Herbert O. Ramp

BASIC-Praxis

einschließlich Mark III-BASIC

Zweite, verbesserte und erweiterte Auflage

Programmieren leicht und schnell erlernbar

R. Oldenbourg Verlag Wien München

FORTBILDUNG DURCH SELBSTSTUDIUM

Herausgeber: Julius Schärf

Die Buchreihe
„FORTBILDUNG DURCH SELBSTSTUDIUM"
verhilft zu einer schnellen und sicheren Aneignung von Wissen, das für den beruflichen Aufstieg in der heutigen Leistungsgesellschaft erforderlich ist.

Anhand von vollständig durchgeführten Musterbeispielen dringt der Leser schrittweise in das Wissensgebiet ein. Viele Übungen mit angegebenen Lösungen geben ihm die Möglichkeit, die erworbenen Kenntnisse zu überprüfen und schon bald die ihm aufgetragenen Arbeiten zu bewältigen.

Prof. Dipl.-Ing. Herbert O. Ramp wurde 1924 geboren, studierte Schwachstromtechnik an der TH Wien, war etwa 15 Jahre in der Industrie tätig, davon 12 Jahre in den USA, und unterrichtet seit 1965 an einer höheren technischen Lehranstalt hochfrequenztechnische Fächer und Programmieren. Seine fast 20jährige Programmierpraxis begann mit gestecktem Programm und Maschinensprache und führte über Assembler, FORTRAN II und FORTRAN IV zu BASIC. Die Einführung von Time-Sharing an höheren technischen Lehranstalten in Österreich ist seiner Initiative zu verdanken.

HERBERT O. RAMP

BASIC-PRAXIS

einschließlich Mark III-BASIC

Zweite, verbesserte und erweiterte Auflage

R. OLDENBOURG VERLAG WIEN MÜNCHEN 1975

Mit Erlaß vom 24. Oktober 1972 (Zl. 106.905—I/4b/72) vom Bundesministerium für Unterricht und Kunst zum Unterrichtsgebrauch für den IV. und V. Jahrgang der Höheren technischen und gewerblichen Lehranstalten zugelassen.

Schulbuch-Nr. **1271**

1. Auflage 1972

© 1975 R. Oldenbourg KG Wien

Das Werk ist urheberrechtlich geschützt. Die dadurch begründeten Rechte, insbesondere die der Übersetzung, des Nachdrucks, der Funksendung, der Wiedergabe auf photomechanischem oder ähnlichem Wege sowie der Speicherung und Auswertung in Datenverarbeitungsanlagen, bleiben, auch bei nur auszugsweiser Verwertung, vorbehalten. Werden mit schriftlicher Einwilligung des Verlags einzelne Vervielfältigungsstücke für gewerbliche Zwecke hergestellt, ist an den Verlag die gesetzlich zu zahlende Vergütung zu entrichten, über deren Höhe der Verlag Auskunft gibt.

Druck: R. Spies & Co., Wien

ISBN 3-7029-0072-1 R. Oldenbourg Wien
ISBN 3-486-39222-0 R. Oldenbourg München

Inhalt

Vorwort zur ersten und zur zweiten Auflage 7

1. Allgemeines 9

2. Alphamerische Konstanten und Variablen 13
Übungen zu alphamerischen Variablen 19
Lösungen zu den Übungen 20

3. Druckformat 21
Zusammenfassung 26
Übungen zu Druckformat 27
Lösungen zu den Übungen 28

4. Drei- und mehrdimensionale Variablen 29

5. Beispiele (Mark I) 32
5.1 Nicht-eingebaute Kreis- und Hyperbel-Funktionen 32
5.2 Rekursionsformel für Sinus und Cosinus 33
5.3 Zuordnung von Normwerten 36
5.4 Anwendung einer Iterationsformel 37
5.5 Erzeugung Gaußscher Zufallszahlen 39
5.6 Zeichnen einer Kurve 42
5.7 Zeichnen einer Funktion unter Verwendung logarithmischer Teilung an Abszisse und Ordinate 46
5.8 Zeichnen mehrerer Kurven 48
5.9 Zeichnen in Polarkoordinaten 51

6. Aufgaben 1 bis 19 56
Lösungen zu den Aufgaben 71

7. Anhang zu Mark I – BASIC 86
7.1 Überblick über Fehlermeldungen 86
7.2 BCD-Code (Oktal) 88
7.3 Numerische Konstanten 88

8. Mark III – BASIC 89
8.1 Allgemeines 89
8.2 Zeichen 89
8.3 Code 90
8.4 Konstanten 90
8.5 Variablen 90
8.6 Eingebaute Funktionen 90
8.7 Anweisungen 91
8.8 Systembefehle 96
8.9 Verschiedenes 96
8.10 Demonstration des Ablaufs eines Programms in Mark III 97

9. Beispiele (Mark III) 99
9.1 Demonstration der PRINTUSING-Anweisung und der Zeilenmaske 99
9.2 Demonstration der MARGIN-Anweisung 100
9.3 n-maliges Werfen eines m-seitigen Würfels 101
9.4 Zuordnung von Normwerten 103
9.5 Sortieren numerischer Daten 105
9.6 Lösung eines linearen Gleichungssystems mit Matrizenoperationen 106
9.7 Numerisches Differentieren (Zwei-Punkte-Methode) 109
9.8 Integration mittels Simpson-Regel 110
9.9 Zeichnen von sechs Funktionen am Terminal 118
9.10 Eine Aufgabe 120

10. Anhang zu Mark III – BASIC 122
10.1 ASCII-Code (dezimal) auszugsweise 122
10.2 Einige Konstanten auf 19 signifikanten Stellen 122

11. Index aller Programme 123

Vorwort

Beim Erlernen des Programmierens können zwei Phasen unterschieden werden. In der ersten Phase erlernt man Semantik und Syntax (Wortbedeutung und Satzbau) der Programmiersprache, gleichsam die Rechtschreib- und Grammatikregeln. In der zweiten folgt die Anwendung der Sprache, die logische Verknüpfung der Anweisungen, kurz, die Routine des Programmierens. In der Praxis ist die Grenze zwischen den beiden Phasen nicht scharf, die beiden Phasen greifen ineinander.

Der vorliegende Band widmet sich hauptsächlich der zweiten Phase, also der Praxis des Programmierens. Die Kenntnis der Programmiersprache BASIC und zwar in einem Ausmaß entsprechend dem Stoff im Band Schärf – Programmieren ... BASIC, wird hier vorausgesetzt. Die Programmierpraxis beschränkt sich jedoch nicht auf Anwendung der in diesem Band behandelten Anweisungen, BASIC-Praxis erweitert auch das Anweisungsrepertoire. Das in diesem Band verwendete BASIC entspricht dem General - Electric Betriebssystem Mark I/7.

Wien, im Frühjahr 1972 H. O. Ramp

Vorwort zur zweiten Auflage

Seit der ersten Auflage von Schärf „Programmieren in BASIC" und der ersten Auflage von „BASIC-Praxis", ist eine Erweiterung des General Electric Time-Sharing Service auch in Europa verfügbar geworden. Dieses Betriebs-System wird mit Mark II und in einer weiteren Ausbaustufe mit Mark III bezeichnet.

Um der Zugriffsmöglichkeit zu diesem neuen TS-System Rechnung zu tragen wurde in der Neuauflage ein Kapitel dem in Mark III gültigen BASIC gewidmet. Obwohl grundsätzlich gleich dem in „Programmieren in BASIC" und „BASIC-Praxis" verwendeten Mark I-BASIC, ist es jedoch mit diesem nicht voll kompatibel.

Eine erschöpfende Behandlung von Mark III-BASIC wäre wegen des bedeutend erweiterten Anweisungsrepertoirs zu umfangreich. Es ist jedoch ohne weiteres möglich in einem zusätzlichen Kapitel die wesentlichsten Unterschiede bei den häufiger verwendeten Anweisungen ausreichend zu besprechen. Dieses Kapitel ist, im Geiste von „BASIC-Praxis", durch einige in Mark III-BASIC geschriebene Programme untermauert.

Wien, im Herbst 1974 H.O. Ramp

1. Allgemeines

Astronauten halten sich an eine Prüfliste (checklist) damit alles schön der Reihe nach getan und auch nichts vergessen wird. Eine „checklist" kann auch fürs Programmieren aufgestellt werden. Sie wird besonders dem Anfänger eine Hilfe sein. Mit wachsender Erfahrung verschmelzen dann die einzelnen Schritte zu einem homogenen Ganzen. Oft sind die Schritte nicht klar trennbar, oder es ist notwendig, bereits erledigte Schritte wieder aufzugreifen, wenn sich durch spätere Schritte neue Perspektiven ergeben. Auf keinen Fall ist Programmieren ein starres Erstens, Zweitens, Drittens, . . . , sondern ein dynamisches Wechselwirken zwischen Definition, Methode, Algorithmus, Flußdiagramm, Codierung und Anwendung. Gerade dieses Wechselwirken zwingt den Techniker und Ingenieur, sein eigener Programmierer zu sein.

Hier sieben Schritte zur Erstellung eines Programms:

1) Definition des Problems
   Was ist gegeben, was ist gesucht, was soll ausgedruckt werden, was sind die Grenzwerte, Genauigkeit usw.
2) Bestimmung einer Methode
   Numerische Analyse, Algorithmus, mathematische Aufbereitung des Problems.
3) Mechanisierung der Methode
   Allgemeine Organisation der Schritte, Unterprogramme usw.
4) Flußdiagramm
   Grob/Fein-Festlegung der Schritte, logischer Aufbau des Programmablaufes.
5) Codieren
   Schreiben des Programms, basierend auf dem Flußdiagramm.
6) Compilieren und Prüfen
   Feststellung von sprachlichen (Syntax) und logischen Fehlern, Prüfung der Richtigkeit des Programms mittels eines handgerechneten Beispiels.
7) Endgültiges Prüfen
   Prüfung ob Extremwerte richtig verarbeitet, Prüfung auf Immunität gegenüber falschen, sinnlosen Eingaben.

Die Prüfliste, obwohl eine Hilfe bei Erstellung des Programms, garantiert nicht die Fehlerlosigkeit des Programms. Von den beiden Fehlerarten, Syntax-Fehler und logische Fehler, sind die Syntax-Fehler (Rechtschreibfehler könnte man sie nennen) die weitaus harmloseren – der Compiler [1] stellt in einer Diagnostik fest was falsch ist (z.B. INCORRECT FORMAT IN . . . .). Schwerwiegender sind die logischen Fehler, besonders dann, wenn sie sich nur bei bestimmten Kombinationen von Eingabedaten bemerkbar machen.

---

[1] Unter Compiler versteht man ein computereigenes Programm, welches die problemorientierte Sprache, hier BASIC, in die vom Computer allein verstandene Maschinensprache übersetzt.

In Lehrbüchern über Programmieren, einschließlich dem vorliegenden, werden die Programme in ihrer endgültigen Form, d.h. ohne Syntax-Fehler und (wie der Autor hofft) ohne logische Fehler präsentiert. Das täuscht eine unrealistische Perfektion vor. Es ist praktisch unmöglich, ein Programm ohne Syntax- und/oder logische Fehler zu schreiben. Jedes Programm hat mehrere „Läufe" hinter sich, ehe es eine endgültige Form gefunden hat.

Der Fehlersuche (debugging) kommt deshalb eine ganz wesentliche Bedeutung zu. Es sei daher auch eine Prüfliste gegeben, die bei der Fehlersuche helfen soll (aber nicht das Finden eines Fehlers garantieren kann). Voraussetzung für die Anwendung der Prüfliste ist (praktisch immer) das Vorhandensein eines handgerechneten Beispiels (es ist ratsam dazu einen Tischrechner zu benützen, da Rechenstabgenauigkeit oft nicht ausreicht).

Hier die P r ü f l i s t e:

1) Sind die Daten richtig?
   Wurde bei INPUT richtig eingegeben?
   Stimmen die Konstanten in der DATA-Anweisung? (Man prüfe Wert und Reihenfolge der Daten!)
   Sind die im Programm eingebauten Konstanten korrekt?
2) Sind alle Variablen logisch früher im Programm definiert worden?
   D.h. sind rechts vom Zuweisungszeichen (=) stehende Variable links vom Zuweisungszeichen oder in der Liste von READ oder INPUT gestanden? (Nicht definierten Variablen wird vom Compiler der Wert Null zugeordnet!)
3) Wird bei Verzweigungsanweisungen richtig verzweigt? Stimmt bei IF - THEN Anweisungen das >, =, < so wie verwendet?
   Stimmen bei ON - GOTO oder GOTO die Zeilennummern?
4) Sind die FOR-Schleifen richtig?
   Stimmen Anfangs- und Endwert? Besonders im Zusammenhang mit der Schrittgröße! (Schrittgröße Null kann zu endloser Schleife führen)

Führen die oben genannten Punkte nicht zur Auffindung des Fehlers, dann

5) Einfügen temporärer Druckanweisungen an strategisch wichtigen Stellen des Programms.

Jedes Programm kann logisch unterteilt werden. Ausdrucken an diesen logischen Trennstellen führt zur Eingrenzung der Lage des Fehlers. Ein schrittweises Durcharbeiten des den Fehler enthaltenden Teiles wird meist den Fehler lokalisieren. Bei diesem schrittweisen Durcharbeiten muß man „Computer spielen", d.h. genau das tun was in der Anweisung steht – nicht mehr und nicht weniger. Beim „Computer spielen" scheidet sich Programmierer-Spreu von Programmierer-Weizen.

Ist der Fehler gefunden, fragt sich jeder ernstlich am Programmieren Interessierte sehr bald: „Wie kann ich mein Programm verbessern? ", wobei verbessern heißt: mit einem kürzeren Programm schneller die richtige Antwort erhalten [1].

---

[1] kurz, bzw. lang, bezieht sich auf die physische Länge des Programms, d.h. die Anzahl der Zeichen im Programm; schnell, bzw. langsam, bezieht sich auf die Laufzeit des Programms, d.h., die in BASIC mit USED xx.xx UNITS (früher RAN xx.xx SEC) ausgegebene Zeit.

Allerding ist „kürzer" und „schneller" oft nicht vereinbar. Man betrachte z.B., die Bestimmung, ob eine Zahl Z gerade oder ungerade ist.

```
15Ø X=Z/2                          16Ø IF Z/2=INT(Z/2) THEN 25Ø
16Ø IF X=INT(X) THEN 25Ø
```

Die linke Version ist schneller, denn sie enthält nur eine Division (Z/2 wird einmal berechnet und zweimal verwendet), ist aber länger als die rechte Version. Für die Länge zählen sämtliche Zeichen, also auch Zeilennummern und Blankstellen (!) und zwei Zeichen extra pro Zeile (RETURN, LINE FEED). Die linke Version hat demnach eine Länge von 32 Zeichen (und braucht eine Extra-Speicherstelle für X), die rechte Version eine Länge von 26 Zeichen (Blanks nicht gezählt).

In „batch-processing", bei dem einem der Computer für die Laufdauer des Programms voll und ganz gehört, ist die linke Version wahrscheinlich die bessere. In „time-sharing" (TS), bei dem Speicherplatz kostbar ist und das Einlesen eines Programms doppelt kostet (Telefon und Computeranschlußgebühr), ist die rechte Version eher angebracht.

Doch „kurz" und „schnell" treten eindeutig in den Hintergrund, wenn es sich um die logische Korrektheit des Programms handelt. Der am schwersten wiegende Programmierfehler ist jener, der zur Ausgabe einer falschen Antwort unter dem Deckmantel einer richtigen führt. Beispiel: Ein Programm soll Nullstellen einer gegebenen Funktion in einem gegebenen Intervall bestimmen. Eingabe von $y=x^2$ von -1 bis $+1$ führt zur Ausgabe „keine Nullstellen zwischen -1 und $+1$". Der Fehler lag in der Logik des Programms, die nur auf Vorzeichenwechsel in y prüfte.

In den gleichen Rahmen gehört auch eine Überprüfung von Eingabedaten. Es muß verhindert werden, daß sinnlose Eingaben zu einer scheinbar richtigen Ausgabe führen. Beispiel: Antennenberechnung. Antennendurchmesser 2m, Höhe der Mitte der Antenne über dem Boden Ø,5m. – Eine physikalisch unmögliche Situation.

Und letztlich: Kein Programm, zumindest kein größeres, ist perfekt. Irgendwo kann es schneller, irgendwo kann es kürzer gemacht werden; irgendeine Eingabekombination bringt das Programm in Schwierigkeiten. Es hängt von der Art des Programms, seinem Verwendungszweck (Hilfsprogramm,wird nur einmal laufen; Programm wird nur vom Programmierer selber verwendet; Programm wird einem beschränkten Personenkreis zur Verfügung stehen; Programm wird Bibliotheksprogramm) ab, welcher Aufwand beim Verbessern und Testen getrieben wird.

Die Programmiersprache BASIC wurde (neben anderen Sprachen, wie z.B. FOCAL), speziell für „conversational, interactive time-sharing" entwickelt. „Conversational, interactive" kann frei übersetzt werden als „sich zusammenreden, sich gegenseitig steuern". Die „Konversation" mit dem Computer muß natürlich einer gewissen „Semantik" und „Syntax", d.h. gewissen Rechtschreib- und Grammatikregeln genügen. Die Regeln die dieses „Gespräch" mit dem Computer ermöglichen, sind in den „System-Befehlen" festgelegt. Sie unterscheiden sich von den „Programm-Befehlen" in erster Linie durch das Fehlen der Zeilennummer.

Unterschiede zwischen den verschiedenen Time-Sharing Systemen sind besonders bei diesen Befehlen sehr augenscheinlich. Es ist jedoch weder sinnvoll, noch möglich, auf alle Versionen einzugehen.

Zum Abschluß des allg. Teiles die Erklärung einiger häufig verwendeter Begriffe.

Programmname: Dient zur Identifizierung des individuellen Programms. Der Programmname besteht aus 1 bis 6 beliebigen Zeichen, wobei das erste ein Buchstabe sein muß. Die Verwendung von Blank und $ soll vermieden werden, da besonders $ eine spezielle Bedeutung hat.[1)]

Arbeitsspeicher: Ist der Speicher, in dem sich das Programm zur Ausführung befinden muß. Ein Programm kommt in den Arbeitsspeicher durch Eingabe von Terminal (über Tastatur oder Lochstreifen) oder durch Aufrufen aus dem Bibliotheksspeicher. Änderungen (Zusätze, Streichungen, Korrekturen) können an einem Programm nur dann gemacht werden, wenn es sich im Arbeitsspeicher befindet.

Bibliotheksspeicher: Ist ein Speicher, in dem Programme – allgemein Dateien – für spätere Verwendung aufgehoben werden können. Ein Programm kommt durch System-Befehle vom Arbeitsspeicher in den Bibliotheksspeicher, bzw. vom Bibliotheksspeicher zurück in den Arbeitsspeicher.

Systembibliothek: Ein Speicher, in dem sich für alle Teilnehmer zugängliche (aber von ihnen nicht veränderbare) Programme befinden. Diese Programme sind durch *** in der 7., 8. und 9. Stelle des Namens gekennzeichnet.

[1)] $, oder auch * an 6ter Stelle des Programmnamens, verhindern unautorisiertes Handhaben des Programms.

2. Alphamerische Konstante und Variable

Eine Programmausgabe ohne erklärenden Text hat nur sehr beschränkten Wert. Dieser Tatsache trägt BASIC Rechnung. Es erlaubt auf einfachste Art Erklärungen in das Programm einzubauen.

Z.B.
```
 1Ø PRINT "TABELLE 8"
 2Ø PRINT "X", "X↑2", "X↑3", "X↑(1/2)", "1/X"
23Ø PRINT "ANZAHL DER MESSWERTE="; N
```

In Anweisung 1Ø wird eine Tabelle näher bezeichnet; in Anweisung 2Ø werden Köpfe für fünf später zu berechnende Zahlenkolonnen gedruckt; 23Ø sagt aus, was die in dieser Zeile befindliche Zahl (N) bedeutet.

Zwischen den Anführungszeichen können alle möglichen Zeichen – Buchstaben, Ziffern, Sonderzeichen – stehen, und man bezeichnet alle solcherart textmäßig verwendeten Zeichen als alphamerische Zeichen (Akronym aus alphabetisch und numerisch). Die in BASIC zugelassenen Zeichen sind:

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Ø 1 2 3 4 5 6 7 8 9
# $ ' ( ) * : - = [ + ; ↑ ] , < . > ? /
```
[1)]

Das von Anführungszeichen Eingegrenzte bezeichnet man als alphamerische Konstante [2)]. Der Wert der Konstanten, d.h. die Art der Zeichen und ihre Reihenfolge, kann vom Programm her nicht geändert werden.

(Man erinnere sich an numerische Konstante. Z.B. U=6.28318531*R.
6.28318531 ist als numerische Konstante vom Programm her nicht veränderbar).

Damit man auch Textausgabe vom Programm her steuern kann, wurden in BASIC alphamerische Variable eingeführt. Eine alphamerische Variable wird mit einem Buchstaben, gefolgt von $ - Zeichen bezeichnet – daher auch die Bezeichnung $ - Variable. $ - Variable dürfen auch indiziert werden, aber nur einfach (bis Index 1Ø automatisch dimensioniert; für >1Ø : 5 DIM V$(15), z.B.)

Beispiele: P$, M$, A$(H), R$(0)
Illegal sind: A2$, B, C(K,L)

[1)] Beim Ausdrucken vom Computer her wird: ! zu . % und \ zu # & zu +
Grundsätzlich nicht ausdruckbar ist @ und ← als Korrekturzeichen.

[2)] Daher ist ", weil Begrenzungszeichen für alphamerische Konstante, normal nicht ausdruckbar.

Mit $-Variablen ist nur eine beschränkte Anzahl von Operationen zugelassen und zwar:

Wertzuweisung mit LET	1Ø0 A$=B$
(implizit)	11Ø C$="5"
	12Ø D$="TOLERANZ-GRENZE"
mit Read	13Ø READ E$, F$(I)
	14Ø DATA "3TER", LETZTER
mit INPUT	17Ø INPUT G$
Ausgabe (siehe auch PRINTUSING)	18Ø PRINT H$(K), I$; J$
Vergleich	19Ø IF K$ < = L$ THEN 5Ø0
	2Ø0 IF M$ = "BASIC" THEN 6Ø0

(Es sind alle 6 Bedingungen zugelassen)

Verglichen werden die $-Variablen in BCD-Darstellung

(d.h. B>A; *>$>R etc.[1] Oder PASS < PASSIV; PASSIV > PASSABEL)

Mit $-Variablen können keine arithmetischen Operationen durchgeführt werden.

Z.B. 1Ø A$=B$ + C$ ist illegal

Daher darf eine $-Variable auch nicht als Schleifenvariable verwendet werden.

Z.B. 1Ø FOR A$ = "BEG" TO "END" ist illegal

Zur Demonstration der Anwendung von $-Variablen diene folgendes Beispiel: An einem Gerät werden Messungen vorgenommen. Die Meßwerte werden in einem Programm verarbeitet und nach Auswertung wird ein Parameter, eine Eigenschaft des Gerätes, als unter-, inner- oder oberhalb eines gegebenen Toleranzbereiches deklariert. Neben dem errechneten Parameter soll ein entsprechender Text ausgedruckt werden.

Im Programmbeispiel werden Zufallszahlen erzeugt und festgestellt wo die jeweilige Zahl bezüglich zweier Schranken liegt. Es sind drei Lösungsmöglichkeiten gegeben.

1. Möglichkeit. Es werden nur $-Konstante verwendet.

Besprechung:
1Ø0,11Ø...Einlesen der Grenzwerte
12Ø,13Ø...Ausdrucken von Text und Grenzwerten

[1] Siehe BCD-Code im Anhang (BCD = Binary Coded Decimal)

15Ø...Initialisierung des Zufallszahlengenerators
16Ø...27Ø...Schleife für die 1Ø zu untersuchenden Zahlen
17Ø...Erzeugung einer Zufallszahl
18Ø...Ausdrucken dieser Zufallszahl und Stehenbleiben in der Zeile
19Ø,2ØØ...Verzweigen, wenn Zufallszahl außerhalb Grenzen
21Ø,23Ø,25Ø...Ausdrucken der zutreffenden Feststellung und Stehenbleiben unmittelbar danach
26Ø...für alle Fälle (ober-, inner-, unter-halb) gemeinsamer Anschlußpunkt. Nach Ausdrucken Weiterschalten in neue Zeile
29Ø...Sprung zum Einlesen neuer Grenzwerte

```
RA#1        10:04    WIEN 02/06/71

100 READ W1,W2
110 DATA 3.5,6.5
120 PRINT "UNTERE GRENZE ";W1
130 PRINT " OBERE GRENZE ";W2
140 PRINT
150 W=RND(-1)
160 FOR I=1 TO 10
170 W=10*RND(0)
180 PRINT W,
190 IF W<W1 THEN 230
200 IF W>W2 THEN 250
210 PRINT "INNERHALB";
220 GOTO 260
230 PRINT "UNTERHALB";
240 GOTO 260
250 PRINT " OBERHALB";
260 PRINT " TOLERANZ-BEREICH"
270 NEXT I
280 PRINT
290 GOTO 100
```

```
RA#1        10:02    WIEN 02/06/71

UNTERE GRENZE   3.5
 OBERE GRENZE   6.5

 5.18761         INNERHALB TOLERANZ-BEREICH
 6.1071          INNERHALB TOLERANZ-BEREICH
 7.80747          OBERHALB TOLERANZ-BEREICH
 9.58402E-02     UNTERHALB TOLERANZ-BEREICH
 4.02934         INNERHALB TOLERANZ-BEREICH
 1.9151          UNTERHALB TOLERANZ-BEREICH
 7.31025          OBERHALB TOLERANZ-BEREICH
 5.02192         INNERHALB TOLERANZ-BEREICH
```

```
 1.10723          UNTERHALB TOLERANZ-BEREICH
 2.87332          UNTERHALB TOLERANZ-BEREICH

OUT OF DATA  IN 100



USED      1.67 UNITS.
```

2.Möglichkeit. Es werden $-Konstante und $-Variable verwendet. Wertzuteilung für die (indizierten) $-Variablen durch implizites LET (d.h. =)

Besprechung:
(siehe auch RA#1)
100,110,120...Zuteilung von Werten an die drei $-Variablen T$(0), T$(1), T$(2)
210,230,250...dem Index I0 werden Werte zugeordnet
260...gemeinsamer Anschlußpunkt für alle Fälle. Ausdrucken der gesamten Zeile in einer Anweisung

```
RA#2        10:07    WIEN 02/06/71

100 T$(0)="UNTERHALB"
110 T$(1)="INNERHALB"
120 T$(2)=" OBERHALB"
130 READ W1,W2
140 DATA 3.5,6.5
150 PRINT "UNTERE GRENZE ";W1
160 PRINT " OBERE GRENZE ";W2
170 PRINT
180 W=RND(-1)
190 FOR I=1 TO 10
200 W=10*RND(0)
210 I0=0
220 IF W<W1 THEN 260
230 I0=1
240 IF W<=W2 THEN 260
250 I0=2
260 PRINT W,T$(I0);" TOLERANZ-BEREICH"
270 NEXT I
280 PRINT
290 GOTO 130
```

```
RA#2       10:06    WIEN 02/06/71

UNTERE GRENZE  3.5
 OBERE GRENZE  6.5

 1.32042        UNTERHALB TOLERANZ-BEREICH
 2.23992        UNTERHALB TOLERANZ-BEREICH
 3.94028        INNERHALB TOLERANZ-BEREICH
 6.22865        INNERHALB TOLERANZ-BEREICH
 .162153        UNTERHALB TOLERANZ-BEREICH
 8.04791         OBERHALB TOLERANZ-BEREICH
 3.44306        UNTERHALB TOLERANZ-BEREICH
 1.15473        UNTERHALB TOLERANZ-BEREICH
 7.24004         OBERHALB TOLERANZ-BEREICH
 9.00613         OBERHALB TOLERANZ-BEREICH


OUT OF DATA  IN 130



USED      1.83 UNITS.
```

3. Möglichkeit. Es werden nur $-Variable verwendet. Wertzuteilung durch READ/DATA.

```
RA#3A      10:36    WIEN 03/05/71

100 FOR I=0 TO 5
110 READ T$(I)
120 NEXT I
130 DATA UNT,INN," OB",ERHALB TOLERANZ,-BEREICH,ERE GRENZE
140 READ W1,W2
150 DATA 3.5,6.5
160 PRINT T$(0);T$(5);W1
170 PRINT T$(2);T$(5);W2
180 PRINT
190 W=RND(-1)
200 FOR I=1 TO 10
210 W=10*RND(0)
220 A=SGN(W1-W)
230 B=SGN(W2-W)
240 PRINT W,T$(INT(ABS((A+B)/2-A*B)+.5));T$(3);T$(4)
250 NEXT I
260 PRINT
270 GOTO 140


RA#3A       7:50    WIEN 21/05/71

UNTERE GRENZE  3.5
 OBERE GRENZE  6.5

 2.94152        UNTERHALB TOLERANZ-BEREICH
 3.86101        INNERHALB TOLERANZ-BEREICH
 5.56138        INNERHALB TOLERANZ-BEREICH
 7.84975         OBERHALB TOLERANZ-BEREICH
 1.78325        UNTERHALB TOLERANZ-BEREICH
 9.66901         OBERHALB TOLERANZ-BEREICH
 5.06416        INNERHALB TOLERANZ-BEREICH
 2.77582        UNTERHALB TOLERANZ-BEREICH
 8.86114         OBERHALB TOLERANZ-BEREICH
 .627223        UNTERHALB TOLERANZ-BEREICH

OUT OF DATA  IN 140



USED      2.00 UNITS.
```

Besprechung:
(siehe auch RA#1)
1Ø0...13Ø...Zuteilung von Werten für das $-Feld T$(I)
16Ø,17Ø...Kopf und Grenzen (entspricht 12Ø, 13Ø in RA#1)
22Ø,23Ø...A und B können −1, Ø oder +1 werden. Zur Verwendung in der Indexberechnung in 24Ø
24Ø...Ausdrucken einer Zeile (entspricht 26Ø in RA#2)

Folgende Regeln für $-Variable sind in diesen Programmen demonstriert:

Die maximale Länge einer $-Variablen oder -Konstanten ist 15 Zeichen (z.B. RA# 3A T$ (3)) (Beachte jedoch: In PRINT darf eine $-Konstante eine ganze Zeile füllen).

Bei Wertzuweisung mit = (implizites LET) müssen Anführungszeichen verwendet werden. Bei Wertzuweisung über READ können Anführungszeichen weggelassen werden, vorausgesetzt die $-Konstante beginnt nicht mit Blank, Ziffer oder Punkt und enthält keinen Beistrich. Im Zweifelsfalle ist Verwendung von Anführungszeichen immer zulässig.

Eine lückenlose Aneinanderreihung von $-Variablen beim Ausdrucken ist durch Setzung eines Strichpunktes möglich (z.B. RA# 3A, Anweisungen 16Ø und 17Ø). Ein Beistrich hat die gleiche Wirkung (15er Feld) wie beim Ausdrucken numerischer Größen.

Am häufigsten finden $-Variable in Kurvenzeichenprogrammen Anwendung.

Übungen zu alphamerischen Variablen

1) Der $-Variablen Q$ ist der Wert GUETE zuzuordnen. Wertzuweisung soll über a) LET (implizit) b) READ c) INPUT erfolgen.

2) Wie 1), aber der zuzuordnende Wert sei 2TER.

3) Das Wort PROGRAMMIERSPRACHE soll als $-Variable zur Verfügung stehen. Eingabe über READ. Name der Variablen beliebig.

4) Ein Programm enthält die Anweisung 1ØØ INPUT F$. Wird MIT eingetippt, soll das Programm nach 3ØØ springen; wird OHNE eingetippt, nach 5ØØ. Für jede andere Eingabe soll das Programm stoppen.

Lösungen zu den Übungen

1) a) 1Ø Q$ = "GUETE"
   b) 1Ø READ Q$
      2Ø DATA GUETE
   c) 1Ø INPUT Q$? GUETE

2) a) 1Ø Q$ = "2TER"
   b) 1Ø READ Q$
      2Ø DATA "2TER"
   c) 1Ø INPUT Q$? "2TER"

3) 1Ø READ V$(Ø), V$(1)
   2Ø DATA PROGRAMMIER, SPRACHE
   oder
   2Ø DATA PRO, GRAMMIERSPRACHE
   oder
   2Ø DATA PROGRAMMIERSPRA, CHE
   oder andere Trennung mit nicht mehr als 15 Zeichen pro Teil.

4) 1ØØ INPUT F$
   11Ø IF F$ = "MIT" THEN 3ØØ
   12Ø IF F$ = "OHNE" THEN 5ØØ
   13Ø STOP

3. Druckformat

Die Programmierung des zentralen Problems, z.B. die Auswertung einer Gleichung, ist meist relativ einfach (abgesehen von der mathematischen Aufbereitung). Einen weit größeren Programmieraufwand erfordert die Anordnung der Ausgabe von Daten und Ergebnissen, von allgemeinem Text, wie "TESTPROBLEM", von speziellen Meldungen, wie "KEINE LÖSUNG" etc.

Mit der PRINT-Anweisung bietet BASIC eine einfache Anweisung zur Handhabung der Ausgabe. Bei dieser Anweisung werden Details bezüglich der Wahl des Druckformats dem Compiler überlassen.

Es gibt drei Formate für das Ausdrucken von Zahlen:

Ganzzahlen-Format:	Die Zahl enthält keinen Dezimalpunkt. Z.B. 9
Fixkomma-Format:	Die Zahl enthält einen Dezimalpunkt. Z.B. 8.7 Dieses Format entspricht der normalen Schreibweise von Dezimalzahlen.
Gleitkomma-Format:	Die Zahl enthält einen Dezimalpunkt und einen Exponentenfaktor. Z.B. 5.1E + Ø3 Das entspricht in normaler Schreibweise $5{,}1 \cdot 1Ø^3$

Beim Ausdrucken von Tabellen und Zahlenkolonnen kann es bei Verwendung von PRINT zu Schönheitsfehlern kommen, besonders dann, wenn sich die Zahlenwerte über mehrere Dekaden erstrecken, weil das vom Compiler gewählte Druckformat von der Größe der Zahl abhängt.

Im besonderen gilt bei PRINT folgendes:

Ganze Zahlen mit einem Betrag bis etwa $1Ø^9$ werden im Ganzzahlen-Format ausgedruckt.[1)]

Z.B. 123456789, Ø, −29 etc. (1 bis maximal 1Ø Ziffern; kein Dezimalpunkt)

Platzbedarf: 2 bis 11 Druckpositionen.[2)]

Nichtganze Dezimalzahlen mit Betrag zwischen $1Ø^{-6}$ und $1Ø^5$ (einschließlich) werden in Fixkomma-Format ausgedruckt.

Z.B. −.ØØØØ12, 1., −98.7654 etc. (1 bis maximal 6 Ziffern und ein Dezimalpunkt)

Platzbedarf: 3 bis 8 Druckpositionen.[2)]

[1)] Genau: $2^{3Ø}-1 = 1Ø73741823$

[2)] Zusätzliche, vom Compiler für Zwischenraum erzeugte Leerstellen nicht gerechnet.

Nichtganze Zahlen mit Betrag $<1Ø^{-6}$ und $>1Ø^{5}$, sowie ganze Zahlen mit Betrag $>1Ø^{9}$, werden im Gleitkomma-Format ausgegeben.

Z.B. 1.23456E + Ø8, −9.87654E−12 etc. (− oder Blank, Ziffer ≠Ø, Dezimalpunkt, 5 Ziffern, E, + oder −, 2 Ziffern)

Platzbedarf: 12 Druckpositionen.[2)]

Damit der Programmierer voll und ganz die Form der Ausgabe bestimmen kann, wurde in BASIC die PRINTUSING-Anweisung eingeführt.

```
1Ø PRINTUSING 2Ø, Liste
2Ø: Zeilenbild (Maske)
```

wobei die (nicht ausführbare) Anweisung 2Ø: als Zeilenmaske bezeichnet wird. (Der Doppelpunkt muß unmittelbar nach der letzten Ziffer der Zeilennummer stehen).

So wie READ mit DATA und FOR mit NEXT, ist die Anweisung PRINTUSING mit der Zeilenmaske : untrennbar verbunden.

In der Zeilenmaske wird, Druckposition für Druckposition, festgelegt, wo in der Zeile Zahlen stehen (und mit welchem Format sie ausgedruckt werden sollen), wo Text steht und wo Leerstellen sind (d.h. alphamerische Zeichen).

In der Liste von PRINTUSING wird festgelegt, welche Zahlenwerte und welcher Text in welcher Reihenfolge in die Maske eingesetzt werden sollen.

Für die Zeilenmaske gilt folgender Code:

bedeutet eine Ziffernstelle (bzw. auch Platz für Vorzeichen)

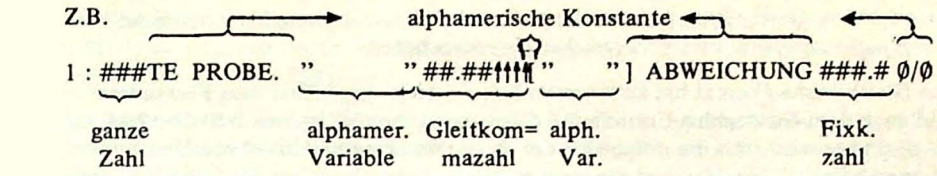
Ganzzahlen-Format:	####	z.B. eine dreistellige ganze Zahl
Fixkomma-Format:	###.##	z.B. eine vierstellige Zahl mit zwei Dezimalstellen
Gleitkomma-Format:	##.### ↑↑↑↑	z.B. eine vierstellige Zahl mit drei Dezimalstellen (die vier Pfeile stehen für: Exponent-Vorzeichen-Ziffer-Ziffer)

Man beachte! Es muß immer auch ein Platz für das Vorzeichen vorgesehen werden.

' bedeutet eine einstellige alphamerische Variable.

" " bedeutet erste und letzte Stelle einer mehrstelligen alphamerischen Variablen. (Die Länge der Variablen ist gegeben durch: Anzahl der Leerstellen zwischen den Anführungszeichen + 2 Stellen. Maximale Länge 15 Stellen)

Sonstige Zeichen in der Zeilenmaske werden als alphamerische Konstante behandelt und wie gegeben ausgedruckt.



Angewendet in einem Programm:

```
RA#F1      10:02    WIEN 31/08/71

1:###TE PROBE.  "         "##.##↑↑↑↑["""]  ABWEICHUNG ###.# Ø/Ø
10 READ N,W,US,D
20 DATA 17,1357.7531,GR,-2.4680864
30 PRINTUSING 1, N,"GEWICHT",W,US,D
40 PRINT
50 READ N,W,US,D
60 DATA 5,963.03696,MG,11.223344
70 PRINTUSING 1, N,"SI-GEHALT",W,US,D

RA#F1      10:02    WIEN 31/08/71

 17TE PROBE.  GEWICHT     1.36E+03[GR]  ABWEICHUNG  -2.5 Ø/Ø

  5TE PROBE.  SI-GEHALT   9.63E+02[MG]  ABWEICHUNG  11.2 Ø/Ø
```

Man beachte! Die Reihenfolge der Anführung der Variablen in der Liste von PRINT USING bestimmt die Reihenfolge der Einsetzung der Werte in die Zeilenmaske. Zahlen müssen mit Zahlenformat (#) und alphamerische Größen mit alphamerischem Format (' oder " ") korrespondieren. Trennung der Eintragungen in der Liste beliebig durch Beistrich oder Strichpunkt (siehe z.B. 7Ø....W;...). Aber Beistrich nach Zeilenmaskennummer! Ein Komma oder Strichpunkt nach der letzten Listeneintragung unterdrückt den Vorschub in eine neue Zeile (Wirkung wie Strichpunkt bei PRINT).

PRINTUSING bewirkt automatisch eine Rundung der letzten ausgedruckten Stelle einer Zahl. Diese Rundung ist aber nur bei Fixkomma- und Gleitkomma-Format wirksam. Bei Ganzzahlen-Format wird nicht gerundet, sondern die Stellen rechts vom Dezimalpunkt werden abgeschnitten.

Beispiel:

```
RA#F2        9:09    WIEN Ø3/Ø9/71

1  Z=123.96503
5:##############.##########.#↑↑↑↑↑↑↑
3Ø PRINTUSING 5, Z,10*Z,100*Z


RA#F2        9:10    WIEN Ø3/Ø9/71

*  123******  1239.7******  12.4E+Ø3***
```

* markiert in diesem Programm geplante Leerstellen, um Leerstellen entstehend durch nicht ausgedrucktes + Vorzeichen, hervorzuheben.

Das Gleitkomma-Format hat zwei wesentliche Vorteile gegenüber dem Fixkomma- und auch dem Ganzzahlen-Format. Die Genauigkeit beträgt immer n Stellen (n=1 bis 8) und es wird stets die gleiche (in der Maske festgelegte) Anzahl von Druckpositionen belegt (7 bis 14), unabhängig vom tatsächlichen Wert der Zahl. Bei Fixkomma-Format müßte man etwa 76 Stellen vor und nach dem Dezimalpunkt vorsehen, wollte man alle vom Computer darstellbaren Zahlen handhaben; eine unmögliche Forderung. Man muß sich also, bei Fixkomma- und Ganzzahlen-Format, in der Zeilenmaske auf eine bestimmte Anzahl von Stellen festlegen.

Was passiert nun bei Fixkomma- oder Ganzzahlen-Format, wenn die Zahl kleiner oder größer ist als in der Maske angenommen?

Ist die Zahl kleiner als in der Maske angenommen, dann werden nur die äußersten rechten Stellen besetzt, eventuell Ø ausgedruckt. Führende Nullen werden unterdrückt und ein Minuszeichen wird unmittelbar vor die erste Stelle gesetzt (z.B. vor den Dezimalpunkt). Die Feldlänge bleibt gleich der Feldlänge in der Maske.

Ist die Zahl größer als erwartet, dann werden soviele Stellen wie benötigt hinzugefügt und das folgende Zeilenbild um diese zusätzlichen Stellen nach rechts gerückt. Die Feldlänge wird also größer als die Feldlänge in der Maske. Das kann zu Übereinanderdrucken in der letzten Druckposition führen.

Bei alphamerischen Zeichen existiert ein ähnliches Feldgrößenproblem. Es gilt hier folgendes: Sind weniger oder gleichviel Zeichen vorhanden als Platz in der Maske vorgesehen, dann wird, links adjustiert, alles ausgedruckt. Sind mehr Zeichen vorhanden, dann werden nur die äußersten linken Zeichen, entsprechend dem vorgesehenen Platz, ausgedruckt. Die Feldlänge ist immer gleich der Feldlänge in der Maske.

Ein Beispiel (* als Füllzeichen für geplante Leerstellen):

```
RA#F3      11:05    WIEN 31/08/71

10 Z=2.718
20 V$="BASIC"
30 PRINTUSING 40, Z,-Z,Z,-Z,V$,"FORMAT",10*Z,V$,10*Z
40:##.####*NEG*##.##*##*##.*"     ""     "*##.##↑↑↑↑*'*##.
```

```
RA#F3      11:05    WIEN 31/08/71

 2.718*NEG*-2.72* 2*-3.*BASIC FORMAT* 2.72E+01*B* 27.
```

In den bisherigen Beispielen bestand ein eins-zu-eins Zusammenhang zwischen PRINTUSING-Liste und Zeilenmaske, d.h. für jede Eintragung in der Liste gab es ein Feld in der Maske. Sind mehr Eintragungen in der Liste als Felder in der Maske, wird die Maske erneut verwendet; sind weniger Eintragungen vorhanden, werden die

verbleibenden Felder ignoriert. Im besonderen Fall einer Leerliste (z.B. PRINT-USING 4), wird nur der in der Zeilenmaske (4:) befindliche Text gedruckt (siehe z.B. RAPOM 1: und 23Ø).

Im folgenden Beispiel sei demonstriert, was geschieht. 1: ist eine Maske für zwei Zahlen (plus zwei alphamerische Variable). 1Ø verlangt das Ausdrucken von vier Zahlen und 2Ø einfach das Ausdrucken von drei Sternen, wo immer der Schreibkopf gerade steht.

```
RA#Y       11:08    WIEN 31/08/71

1:""=#.##↑↑↑↑    ""=####
10 PRINTUSING 1,"AB",12,"CD",-34,"EF",-56,"GH",78;
20 PRINT "***"
30 PRINTUSING 1,"YZ",90;
40 PRINT "***"
50 PRINT
60 PRINTUSING 1,"AB",12,"CD",-34;
70 PRINTUSING 1,"EF",-56,"GH",78;
80 PRINTUSING 1,"YZ",90
90 PRINT "***"


RA#Y       11:07    WIEN 31/08/71

AB= .12E+02    CD= -34
EF=-.56E+02    GH=  78    ***
YZ= .90E+02    ***

AB= .12E+02    CD= -34    EF=-.56E+02    GH=  78    YZ= .90E+02
***
```

Die Ausgaben zufolge 1Ø und 2Ø sind in den ersten beiden Zeilen. D.h. nachdem die Maske in 1Ø aufgebraucht war (nach Ausdrucken von –34), wurde in eine neue Zeile geschaltet und das Format von 1: wieder benützt. Nun endet 1Ø aber mit einem Strichpunkt, das heißt: "Schreibkopf soll in der Zeile stehen bleiben", und die drei Sterne von 2Ø kommen daher in dieselbe Zeile wie die letzte Ausgabe. Man beachte die drei Leerstellen zwischen 78 und ***. Diese Leerstellen sind, im Druck nicht sichtbar (nur am Lochstreifen), hinter #### in 1:.

Die Ausgabe zufolge 3Ø ist in der dritten Zeile. Die Maske wurde bis zur nächsten (in der Liste nicht vorhandenen) Variablen (im Beispiel ist das eine zweistellige alphamerische Variable, gekennzeichnet in 1: durch " ") aufgebraucht, dann blieb der Schreibkopf stehen (wie in 3Ø durch ; am Ende verlangt) und 4Ø wurde ausgeführt. Die drei Leerstellen zwischen Zahl und *** sind hier die drei Leerstellen zwischen #### und " " in der Maske. D.h. die Ausführung einer PRINTUSING-Anweisung endet immer nach alphamerischen Konstanten, bzw. vor Variablen.

6Ø, 7Ø, 8Ø zeigt, wie man alle fünf Ausgabegruppen mit der gegebenen Maske in eine Zeile bringen kann. Da 8Ø nicht mit Strichpunkt endet, kommen die drei Sterne von 9Ø in eine neue Zeile[1].

---

[1] Anstatt Strichpunkten hätten im Programm RA#Y auch Beistriche (mit gleicher Wirkung) verwendet werden können.

Zusammenfassung

Folgende Formen der Felder in einer Zeilenmaske sind sinnvoll (möglich sind mehr):

Ganzzahlen-Format:	2 bis 11 #
Fixkomma-Format:	2 bis 9 # und ein Dezimalpunkt (Dezimalpunkt nicht vor dem ersten #)
Gleitkomma-Format:	2 bis 9 #, ein Dezimalpunkt und↑↑↑↑ (Dezimalpunkt nicht vor dem ersten #)
Alphamerisches Format:	1 oder mehr ' unmittelbar hintereinander. "" bis " " (keine Leerstelle bis 13 Leerstellen).
Alphamerische Konstante:	1 bis 72 beliebige Zeichen, außer # ' "

Zahlenfelder müssen durch ein Zeichen anders als # oder . getrennt werden (z.B. ##.##### für −1.23−45 ist nicht möglich). Anführungszeichen müssen paarweise auftreten.

Bei Zahlen sind eventuell entstehende Leerstellen links vor den Ziffern; ist die Zahl zu groß, wird das Feld automatisch erweitert. Bei alphamerischen Zeichen sind eventuell entstehende Leerstellen rechts nach den Zeichen; sind zuviele Zeichen vorhanden, wird das Feld nicht erweitert, sondern überschüssige Zeichen werden unterdrückt.

PRINTUSING ermöglicht

geplantes Ausdrucken von Zahlen, durch vom Programmierer selber festlegbare Formate (Ganzzahlen, Fixkomma, Gleitkomma)

das Ausdrucken von bis zu 8 signifikanten Stellen einer Fixkomma- oder Gleitkomma-Zahl. (PRINT gibt maximal 6 Stellen)

Konkrete Beispiele für die Verwendung von PRINTUSING finden sich in den Beispielen RA#NA, RA#E, RA#K, RA#S, RA#A, RAPLO6, RAPOM.

Übungen zu Druckformat

KORRIGIERE FOLGENDE (ZUSAMMENGEHOERIGE) ANWEISUNGEN

1)
```
1:##.##"  "##.#↑↑↑↑***
10 PRINTUSING 1; A,B,"GRAD" , T
```

2)
```
1:'=##.#######↑↑ ↑↑
10 PINTUSING 1, "C", 2.997925E8
```

3)
```
1:''##.#####"   "
10 PRINTUSING 1, "G=",9.80665,M/S↑2
```

4)
```
1: STAND   "      "     "
10 PRINTUSING 1, TS, VS(1,3)
```

5)
```
1:"UEBERSCHRIFT"
10 PRINTUSING 1,
```

6)
```
1: .####NM
10 PRINTUSING 10,.9824
```

Lösungen zu den Übungen

```
1)    1:##.## ##.##"  "##.#↑↑↑↑****          FELD FUER B FEHLTE
      10 PRINTUSING 1, A,B,"GRAD",T          ; NACH 1 WAR ILLEGAL

2)    1:'=##.#####↑↑↑↑                       BLANK IN DEN 4 HOCHPFEILEN
                                             ILLEGAL

3)    1:""##.#####"   "                      "" ANSTELLE VON ''
      10 PRINTUSING 1,"G=",9.80665,"M/S↑2" " " UM M/S↑2 FEHLTE

3A)   1:''##.#####"   "                      UNVERAENDERT, ABER 10
                                             GEAENDERT
      10 PRINTUSING 1,"G","=",9.80665,"M/S↑2"  ZWEI EINSTELLIGE
                                                $-VARIABLE

4)    1: STAND   "      ""    "              ANFUEHRUNGSZEICHEN MUESSEN
                                             PAARWEISE AUFTRETEN
      10 PRINTUSING 1,T$,V$(4)               $-VARIABLE NUR EINDIMEN-
                                             SIONAL ZULAESSIG

5)    1:UEBERSCHRIFT                         EINFACHE ALPHAMERISCHE
                                             KONSTANTE
      10 PRINTUSING 1                        BEI LEERLISTE KEIN KOMMA
                                             NACH ZN, WENN WEITERSCHALTEN
                                             IN NEUE ZEILE ERWUENSCHT

6)    1:#.####NM                             DEZIMALPUNKT DARF NICHT AN
                                             SPITZE EINES ZAHLENFELDES
                                             STEHEN
      10 PRINTUSING 1, .9824                 ZN DER MASKE WAR FALSCH
```

4. Drei- und mehrdimensionale Variable

BASIC erlaubt maximal zwei Indices [1]. Es ist aber ohne weiteres möglich, daß zur Lösung eines Problems mehr Indices benötigt werden. Diese Probleme können in BASIC durch "Linearisierung" gehandhabt werden. Als Beispiel diene ein drei-dimensionaler Raum mit "Grundfläche" 5x4 und "Höhe" 3, so wie in Abb. D1 dargestellt.

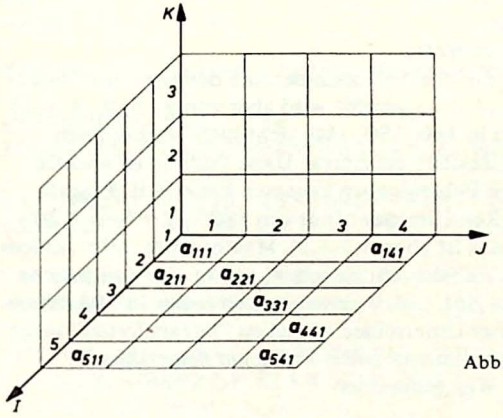


Abb. D1 Symbolische Darstellung von A(I,J,K)

Eine "Linearisierung" des drei-dimensionalen Raumes *I,J,K* ist wie folgt möglich: Nimm die *J*=1 Spalte mit den Zeilen *I*=1...5, füge daran *J*=2 Spalte mit Zeilen *I*=1...5 etc., bis *J*=4 Spalte. Damit ist das "Erdgeschoß", d.i. *K*=1, statt nebeneinander, hintereinander aufgereiht. Setze in gleicher Weise für *K*=2 und *K*=3 fort.

Diese, nun eindimensionale,Anordnung ist numeriert von A(1) für a_{111}, über A(2Ø) für a_{541} bis A(6Ø) für a_{543}. A(2Ø) folgt, weil es 4 5-zeilige Spalten gibt; A(6Ø) folgt, weil es 3 Niveaus à 2Ø Plätze gibt.

Allgemein gilt für 3-dimensionale Variable: Ist *I*Ø die Anzahl der Zeilen, *J*Ø die Anzahl der Spalten und *K*Ø die Anzahl der Niveaus, dann ist die Variable mit *I*Ø*J*Ø.*K*Ø zu dimensionieren. Für Beispiel: 5 · 4 · 3 = 6Ø, d.h. DIMA(6Ø). Das Element *A(I,J,K)* wird angesprochen mit

$$A(I+I\emptyset*(J-1)+I\emptyset*J\emptyset*(K-1))$$

wenn die Indizierung wie in der Abbildung mit 1 beginnt.

---

[1] Andere Sprachen oder Systeme erlauben eine andere Anzahl von Dimensionen. Z.B. Time-Sharing FORTRAN (General Electric) erlaubt 15-dimensionale Variable, FOCAL (Digital Equipment Corporation) nur ein-dimensionale Variable.

Als Beispiel sei folgendes Problem betrachtet: Ein Betrieb erzeugt fünf Artikel. Diese Artikel werden in 4 Arten von Geschäften verkauft (Großkaufhaus, Fachgeschäft etc.). Bei den Geschäften sind drei Lokalitäten zu unterscheiden (Vorort, Zentrum etc.). Für eine Verkaufsstatistik soll, nach Eingabe von Artikelnummer, Geschäftsart und Lage, ausgegeben werden wie viel Stück dieses Artikels im gegebenen Geschäft verkauft wurden. Ist eine der Nummern gleich Null, soll eine Gesamtübersicht ausgegeben werden. (Im Beispiel werden die einzelnen Positionen mit Zufallszahlen gefüllt).

Besprechung des Programms.

30...50...füllt die 60 Speicherplätze mit Zufallszahlen
60...90...$-Variablen werden Werte zugeteilt
120...Eingabe der Daten über Tastatur
130...Verzweigen zum Ausdrucken aller Werte
140...160...hier wird geprüft ob jeder der drei Indices innerhalb der gegebenen Grenzen ist. Bereiche sind 1...5, 1...4, 1...3, geprüft wird aber von 0...5, 0...4, 0...3, da 0 in 130 geprüft wird, also in 140, 150, 160 nicht mehr vorkommen kann. Man kann dadurch drei Zeichen einsparen. Diese Prüfung ist wichtig, da es ohne sie zu irreführenden Fehlausgaben kommen kann. Z.B. Eingabe bei 120 sei: $I=-2, J=6, K=0$. Resultierender Index in 180: $-2+5\cdot 5+20\cdot -1=3$, daher Ausgabe gleich 17 und das ist aber $A(3,1,1)$. Man beachte, alle Indices sind in obigem Beispiel falsch. Es ist leicht zu sehen, daß es eine beachtliche Anzahl von "falschen" Indices gibt, die zu einem legalen Index in 180 führen. Hier macht sich ein wesentlicher Unterschied zwischen "linearisierten" mehrdimensionalen und echten mehrdimensionalen Variablen bemerkbar.
200,210...bei Indexfehler wird dieser Weg genommen
230...350...Gesamttabelle
260...Ausdrucken der Lokalität und A1...A5
280...Ausdrucken der Geschäftsart (und Stehenbleiben in dieser Zeile)
290...310...Ausdrucken von fünf Werten

```
RA#NA       8:40    WIEN 18/05/71

0 DIM A(60)
1:""        A1    A2    A3    A4    A5
2:     ""
3:######
10 PRINT "VERKAUFS-STATISTIK"
20 PRINT "BEISPIEL FUER 3-DIMENSIONALE VARIABLE"
30 FOR L=1 TO 60
40 A(L)=INT(101*RND(0))
50 NEXT L
60 FOR I=1 TO 5
70 READ A$(I),G$(I),L$(I)
80 NEXT I
90 DATA A1,G1,L1,A2,G2,L2,A3,G3,L3,A4,G4,L4,A5,G5,L5
100 PRINT
110 PRINT "ARTIKEL(1...5), GESCHAEFTSART(1...4), LAGE(1..3)";
120 INPUT I,J,K
130 IF I*J*K=0 THEN 220
140 IF SGN(-I)=SGN(5-I) THEN 200
150 IF SGN(-J)=SGN(4-J) THEN 200
160 IF SGN(-K)=SGN(3-K) THEN 200
170 PRINT "VERKAUFT  ";A$(I);" ";G$(J);" ";L$(K);" --- ";
```

```
180 PRINT A(I+5*(J-1)+20*(K-1)),"STUECK"
190 GOTO 100
200 PRINT "BEACHTE GRENZEN FUER EINGABE"
210 GOTO 100
220 PRINT
230 PRINT "GESAMTUEBERSICHT"
240 PRINT
250 FOR K=1 TO 3
260 PRINTUSING 1,L$(K)
270 FOR J=1 TO 4
280 PRINTUSING 2,G$(J);
290 FOR I=1 TO 5
300 PRINTUSING 3,A(I+5*(J-1)+20*(K-1)),
310 NEXT I
320 PRINT
330 NEXT J
340 PRINT
350 NEXT K

RA#NA      13:28    WIEN 02/06/71

VERKAUFS-STATISTIK
BEISPIEL FUER 3-DIMENSIONALE VARIABLE

ARTIKEL(1...5), GESCHAEFTSART(1...4), LAGE(1..3)? 3,2,3
VERKAUFT  A3 G2 L3 ---   22    STUECK

ARTIKEL(1...5), GESCHAEFTSART(1...4), LAGE(1..3)? 2,2,2
VERKAUFT  A2 G2 L2 ---   70    STUECK

ARTIKEL(1...5), GESCHAEFTSART(1...4), LAGE(1..3)? 1,5,1
BEACHTE GRENZEN FUER EINGABE

ARTIKEL(1...5), GESCHAEFTSART(1...4), LAGE(1..3)? 0,2,3

GESAMTUEBERSICHT

L1          A1    A2    A3    A4    A5
     G1     71    33    17     9     7
     G2     21    75     4    56    92
     G3     85    19    96    21    22
     G4     32     0    87    65    19

L2          A1    A2    A3    A4    A5
     G1     49    64    89    56    16
     G2     30    70    20    81    61
     G3     85    88    17    34    11
     G4     35     2    25    25    39

L3          A1    A2    A3    A4    A5
     G1     13    10     3    76    27
     G2     68    21    22    19    74
     G3     57    56    68     3    87
     G4     51    47    34    86    87
USED       7.50 UNITS.
```

5. Beispiele (Mark I)

5.1 Nicht-eingebaute Kreis- und Hyperbel-Funktionen

An trigonometrischen Funktionen sind in BASIC (und fast allen anderen problemorientierten Sprachen) eingebaut: Sinus, Cosinus, Tangens, Arcustangens. Andere trigonometrische Funktionen, im besonderen die hyperbolischen Funktionen, müssen „programmiert" werden.

Die Formeln dafür, aufgebaut aus eingebauten Funktionen, sind:

arcsin:	ATN(X/SQR(1−X*X))
arccos:	ATN(SQR(1−X*X)/X)
arccot:	ATN(1/X)
sinh[1]:	(EXP(X)−1/EXP(X))/2
cosh[1]:	(EXP(X)+1/EXP(X))/2
tanh[1]:	(EXP(2*X)−1)/(EXP(2*X)+1)
coth[1]:	(EXP(2*X)+1)/(EXP(2*X)−1)
arcsinh:	LOG(X+SQR(X*X+1))
arccosh:	LOG(X+SQR(X*X−1))
arctanh:	LOG((1+X)/(1−X))/2
arccoth:	LOG((X+1)/(X−1))/2

Fehlermeldungen für vier spezielle Werte des Arguments

x	0	0,5	1	2
arcsin	– – –	– – –	DIV. BY ZERO	SQUARE ROOT OF NEGATIVE NUMBER
arccos	DIV. BY ZERO	– – –	– – –	
arccot	DIV. BY ZERO	– – –	– – –	– – –
sinh	– – –	– – –	– – –	– – –
cosh	– – –	– – –	– – –	– – –
tanh	– – –	– – –	– – –	– – –
coth	DIV. BY ZERO	– – –	– – –	– – –
arcsinh	– – –	– – –	– – –	– – –
arccosh	SQUARE ROOT OF NEGATIVE NUMBER		– – –	– – –
arctanh	– – –	– – –	DIV. BY ZERO	LOG OF NEG. NR
arccoth	LOG OF NEGATIVE NUMBER		DIV. BY ZERO	– – –

[1] Vorberechnung von EXP(X), bzw. EXP(2*X), ist zur Erhöhung der Geschwindigkeit unter Umständen ratsam.

5.2 Rekursionsformel für Sinus und Cosinus

In vielen technischen und mathematischen Problemen muß ein Winkel zwischen zwei Grenzen in gleichen Schritten erhöht werden. Im Verlaufe der Rechnung wird dann meist der Sinus und/oder Cosinus dieses Winkels benötigt.

Das Problem kann grundsätzlich mit folgender Befehlsfolge gelöst werden: (B Anfangswert, E Endwert, D Inkrement)

```
FOR A=B TO E STEP D
S=SIN(A)
C=COS(A)
.
.
.
NEXT A
```

Diese Methode verlangt aber wiederholtes Aufrufen der SIN- und COS-Funktion und das ist zeitaufwendig. Eine drei- bis viermal schnellere Methode (allerdings aber auch längere) verwendet Rekursionsformeln für Sinus und Cosinus.

Aus: $\sin(a+d) = \sin(a)\cos(d) + \cos(a)\sin(d)$
$\cos(a+d) = \cos(a)\cos(d) - \sin(a)\sin(d)$

wobei a der alte Winkel, d das Winkelinkrement und $a+d$ der neue Winkel ist, folgen direkt Rekursionsformeln. Es müssen im ganzen nur zwei Sinus-Werte und zwei Cosinus-Werte (sin/cos(Anfangswinkel), sin/cos(Winkelinkrement)) bestimmt werden.

Die Schnelligkeit wird auf Kosten der Genauigkeit erreicht, denn die Rekursionsformel ist nicht beliebig oft anwendbar. Abhängig von der geforderten Genauigkeit wird man nach einer gewissen Anzahl von Anwendungen abbrechen, und die Rekursionsformel neu starten müssen.

Mit der Rekursionsformel schaut das Programm etwa wie folgt aus:

Initialisierung
```
CØ=COS(D)
C=COS(B-D)
SØ=SIN(D)
S=SIN(B-D)
```

Anwendung
```
FOR A=B TO E STEP D
T=C*CØ-S*SØ
S=S*CØ+C*SØ
C=T
.
.
.
NEXT A
```

In T wird vorübergehend der neue Cosinus gespeichert, denn der alte Cosinus-Wert wird erst noch für die Berechnung des Sinus benötigt.

Wird nur der Sinus oder Cosinus benötigt, so ist eine weitere Steigerung der Geschwindigkeit möglich, jedoch mit weiterer Reduktion der Genauigkeit. Das Programm könnte in diesem Fall wie folgt aussehen:

```
                                  Sinus                Cosinus
                  {                 C0=2*COS(D)
Initialisierung   {  S1=SIN(B-2*D)        C1=COS(B-2*D)
                  {  S=SIN(B-D)           C=COS(B-D)
                                  FOR A=B TO E STEP D
                  {  T=C0*S-S1            T=C0*C-C1
Anwendung         {  S1=S                 C1=C
                  {  S=T                  C=T
                                  .
                                  .
                                  .
                                  NEXT A
```

Es werden in diesem Fall zwei vorhergehende Werte benötigt.

Im folgenden Beispiel werden 650 Schritte gemacht und der Sinus und Cosinus wird bei jedem 50sten Schritt ausgedruckt. Die angegebene Laufzeit gilt für die 650 Schritte mit Ausdrucken bei jedem 50sten Schritt.

Im ersten Block erfolgt die Berechnung mit den eingebauten Funktionen.
Genauigkeit 6...7 Stellen.
Im zweiten Block wird die gemeinsame Rekursionsformel für sin und cos verwendet.
Genauigkeit etwa 6 Stellen.
Im dritten Block werden die getrennten Rekursionsformeln verwendet.
Genauigkeit 3...4 Stellen.

Eine eingehendere Betrachtung zeigt, daß der Fehlerverlauf vom Anfangswert und auch vom Inkrement abhängig ist. Allgemein hat der Fehler oszillatorischen Charakter mit einem exponentiellen Anstieg der Amplitude.

```
 RA#G          9:11    WIEN 26/05/71

REKURSIONSFORMELN FUER SIN U. COS

SIN U. COS MIT EINGEBAUTER FUNKT.

   W          SIN          COS
  .000      .0000000    1.0000000
  .500      .4794255     .8775826
 1.000      .8414710     .5403023
 1.500      .9974950     .0707372
 2.000      .9092974    -.4161468
```

```
  2.500      .5984723    -.8011435
  3.000      .1411203    -.9899925
  3.500     -.3507829    -.9364568
  4.000     -.7568022    -.6536440
  4.500     -.9775300    -.2107964
  5.000     -.9589245     .2836615
  5.500     -.7055409     .7086692
  6.000     -.2794163     .9601700
  6.500      .2151190     .9765878

LIEF 4.66667 SEK

SIN U. COS GEMEINSAM

    W          SIN          COS
   .000     -.0000000   1.0000000
   .500      .4794256    .8775826
  1.000      .8414710    .5403024
  1.500      .9974951    .0707373
  2.000      .9092976   -.4161468
  2.500      .5984724   -.8011437
  3.000      .1411201   -.9899928
  3.500     -.3507833   -.9364571
  4.000     -.7568028   -.6536440
  4.500     -.9775307   -.2107960
  5.000     -.9589250    .2836623
  5.500     -.7055410    .7086703
  6.000     -.2794159    .9601711
  6.500      .2151200    .9765886

LIEF 1.33333 SEK

SIN U. COS GETRENNT

    W          SIN          COS
   .000     -.0000000   1.0000000
   .500      .4794259    .8775853
  1.000      .8414736    .5403111
  1.500      .9975032    .0707522
  2.000      .9093143   -.4161294
  2.500      .5984977   -.8011324
  3.000      .1411506   -.9899987
  3.500     -.3507546   -.9364886
  4.000     -.7567867   -.6537026
  4.500     -.9775388   -.2108742
  5.000     -.9589648    .2835812
  5.500     -.7056121    .7086071
  6.000     -.2795085    .9601442
  6.500      .2150254    .9766094

LIEF 1.33333 SEK
```

5.3 Zuordnung von Normwerten

In den verschiedenen Zweigen der Technik ist es notwendig, errechnete Werte auf Normwerte zu runden. Eine mögliche Programmierung dieses Problems sei an Hand der "Widerstands-Normreihe" der Nachrichtentechnik demonstriert. Die Werte der Widerstände erstrecken sich über mehrere Dekaden, typisch von 10^0 bis 10^7. Innerhalb jeder Dekade sind 24 Normwerte festgelegt.

Folgender Algorithmus für die Zuordnung eines Normwertes wird im Programm verwendet: Die Normwerte von 1 bis 10 (das sind 25 Werte) sind gespeichert. Der Rechenwert dem ein Normwert zugeordnet werden soll, wird in den Bereich 1 bis 10 skaliert. Nun wird festgestellt zwischen welchen zwei Normwerten der Rechenwert liegt. Die Zuteilung des kleineren oder größeren Normwertes erfolgt durch Vergleich des Quadrates des Rechenwertes mit dem Produkt der benachbarten Normwerte (geometrisches Mittel).

Eine "Normwertzuteilung" würde in einem größeren Programm an mehreren Stellen notwendig sein. Es ist daher angebracht dieses Programm als Unterprogramm zu behandeln. Das Programmbeispiel kann als Testprogramm für das Unterprogramm "Normierung" gedacht werden.

Besprechung:

10...60...dienen zum Einlesen der "Normreihe"
130...Zufallszahlen zwischen 0 und 10^7 werden zur Simulierung der Rechenwerte erzeugt.
150...Verzweigung zum Unterprogramm
900...die 10er-Potenz wird gefunden deren Extraktion X in den Bereich 1...10 bringt.
910...X wird in den Bereich 1...10 gebracht (skaliert)
920...960...es wird gesucht zwischen welchen Normwerten der X-Wert liegt
940...X wird der nächstliegende Normwert zugeteilt und gleichzeitig wieder zurückskaliert (Multiplikation mit W(0))

```
RA#D          8:22    WIEN 25/05/71

10 DIM W(25)
20 FOR I=1 TO 25
30 READ W(I)
40 NEXT I
50 DATA1,1.1,1.2,1.3,1.5,1.6,1.8,2,2.2,2.4,2.7,3,3.3
60 DATA3.6,3.9,4.3,4.7,5.1,5.6,6.2,6.8,7.5,8.2,9.1,10
70 PRINT "ZUORNUNG VON NORMWERTEN"
80 PRINT "FUER ERRECHNETE WERTE"
90 PRINT
100 PRINT "RECHENWERT","NORMWERT"
110 X=RND(-1)
120 FOR I=1 TO 10
```

```
130 X=(10*RND(0))↑INT(7*RND(0))
140 PRINT X,
150 GOSUB 900
160 PRINT X
170 NEXT I
180 STOP
900 W(0)=10↑INT(.434294482*LOG(X))
910 X=X/W(0)
920 FOR W=1 TO 25
930 IF X>W(W+1) THEN 960
940 X=W(W+INT(X*X/(W(W)*W(W+1))))*W(0)
950 RETURN
960 NEXT W


RA#D        8:21    WIEN 25/05/71

ZUORNUNG VON NORMWERTEN
FUER ERRECHNETE WERTE

RECHENWERT      NORM WERT
 97.4422          100
 141833.          150000
 1                1
 304.049          300
 .521295          .51
 39186.2          39000.
 148.626          150
 772530.          750000
 1                1
 67.9531          68.



USED        2.50 UNITS.
```

5.4 Anwendung einer Iterationsformel

Nach Newton-Raphson kann die Lösung einer Gleichung der Form $f(x)=0$ durch folgende Iterationsformel gefunden werden:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

Soll die Quadratwurzel einer Zahl R gefunden werden, dann ist

$$f(x) = x^2 - R = 0 \text{ und es folgt}$$

$$x_{k+1} = 0{,}5 \cdot (x_k + \frac{R}{x_k})$$

Die Konvergenz dieses Verfahrens ist für sehr kleine und auch sehr große Werte von R schlecht. Es ist daher vorteilhaft den Radikanden durch Extraktion einer geraden Zehnerpotenz in den Bereich 1...100 zu bringen.

Besprechung:

60...je nachdem ob R>, = oder <0 ist ergibt SGN(R)+2 1, 2 oder 3 und damit Verzweigung zu 90, 70 oder 110

90...Radikand wird positiv gemacht

110...Zehnerpotenz für Reduzierung in den Bereich 1...100 wird errechnet und halbiert (Quadratwurzel)

120...Radikand wird in den Bereich 1...100 gebracht

130...Anfangswert für Iterationsformel

140...180...wiederholte Anwendung der Iterationsformel

150...die Iterationsformel

160...Prüfung, ob gewünschte Genauigkeit (10^{-7}) erreicht; wenn ja, Sprung aus der Schleife

170...der eben berechnete X-Wert wird der alte X-Wert

180...zurück zur neuerlichen Anwendung der Iterationsformel

190...für den Fall, daß nach 10 Iterationen die gewünschte Genauigkeit noch immer nicht erreicht ist, wird diese Tatsache ausgedruckt[1)]

```
RA#E          8:26    WIEN 25/05/71

1:WURZEL(##.####↑↑↑↑)=##.####↑↑↑↑
10 PRINT "QUADRAT-WURZEL NACH NEWTON"
20 PRINT "X(K+1)=1/2*[X(K)+R/X(K)]"
30 PRINT
40 READ R
50 DATA 2,4E70,-64,0,16E-70
60 ON SGN(R)+2 GOTO 90,70,110
70 PRINTUSING 1,0,0
80 GOTO210
90 R=ABS(R)
100 PRINT "WURZEL AUS BETRAG"
110 P=INT(.5*INT(.434294482*LOG(R)))
120 R1=R/100↑P
130 X0=.5+.5*R1
```

[1)] Durch Reduzierung in den Bereich 1...100 ist die Konvergenz gut,und die Schleife wird vor 10 Iterationen verlassen. Für den allgemeinen Fall der Iteration ist es aber angebracht, die Iteration nach maximal 10 Anwendungen abzubrechen, denn ist dann die gewünschte Genauigkeit noch immer nicht erreicht, so ist das Verfahren schlecht angewendet.

```
140 FOR I=1 TO 10
150 X1=.5*(X0+R1/X0)
160 IF ABS((X0-X1)/X1)<=1E-7 THEN 200
170 X0=X1
180 NEXT I
190 PRINT "NACH 10 ITER. FEHLER >1E-7"
200 PRINTUSING 1,R,X1*10↑P
210 PRINT
220 GOTO 40


RA#E        8:25    WIEN 25/05/71

QUADRAT-WURZEL NACH NEWTON
X(K+1)=1/2*[X(K)+R/X(K)]

WURZEL( 2.0000E+00)= 1.4142E+00

WURZEL( 4.0000E+70)= 2.0000E+35

WURZEL AUS BETRAG
WURZEL( 6.4000E+01)= 8.0000E+00

WURZEL( 0.0000E+00)= 0.0000E+00

WURZEL( 1.6000E-69)= 4.0000E-35


OUT OF DATA  IN 40



USED      1.83 UNITS.
```

5.5 Erzeugung Gaußscher Zufallszahlen

Die in BASIC eingebaute RND-Funktion gibt eine Zufallszahlenfolge mit uniformer Verteilung im Bereich Ø bis <1. Eine allgemeine uniforme Verteilung wird mathematisch beschrieben durch die Gleichung

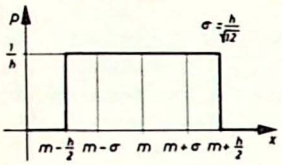
$$p = \frac{1}{h} \cdot \mathrm{rect}\left(\frac{x-m}{h}\right)$$

h...Breite des Bereiches
m...Mitte des Bereiches

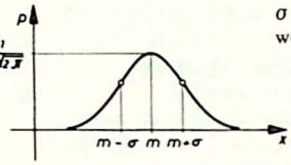
Eine andere Verteilung von großer Bedeutung ist die normale oder Gaußsche Verteilung. Sie wird mathematisch beschrieben durch die Gleichung

$$p = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-m}{\sigma}\right)^2\right)$$

Die in den Gleichungen enthaltenen Größen m und σ sind wesentliche Parameter für Zufallszahlen. Sie werden Mittelwert (m) und Varianz (σ^2) genannt. σ, die positive Wurzel aus der Varianz, wird Standard-Abweichung oder mittlere Schwankung genannt.



Uniforme Verteilung



Normale (Gauß-) Verteilung

σ = Abszisse des Wendepunktes, wenn m = 0

Eine normal verteilte Variable kann aus einer uniform verteilten Variablen u_i (wie RND sie liefert) z.B. durch folgende Beziehung erzeugt werden

$$g = \frac{\sum_{i=1}^{k} u_i - k/2}{\sqrt{k/12}}$$

Im besonderen hat diese Variable den Mittelwert $\emptyset$ (wegen $-k/2$) und die Varianz 1 (wegen $\sqrt{k/12}$). Ist ein anderer Mittelwert oder Varianz erwünscht, dann gilt

$$g' = g \cdot \sigma + m$$

Das Programm soll Zufallszahlen erzeugen und durch Bildung von Mittelwert und Varianz die Verteilung prüfen. Es soll auch der erreichte Minimal- und Maximalwert pro Durchgang bestimmt werden.

Formel für Mittelwert: $$m = \frac{1}{n}\sum_{i=1}^{n} r_i$$

Formel für Varianz: $$\sigma^2 = \frac{1}{n-1}\left[\sum_{i=1}^{n} (r_i)^2 - \frac{1}{n}\left(\sum_{i=1}^{n} r_i\right)^2\right]$$

Wobei r_i beliebige Zahlen darstellt.

Besprechung:

60...N bestimmt die Anzahl der Zahlen, die zur Bildung von Mittelwert und Varianz verwendet werden

70...K ist die in der Formel für die Gaußschen Zufallszahlen auftretende Konstante (12 gibt einfache Rechnung, aber im Programm wurde K für Testzwekke leicht änderbar gemacht)

80,90...später wiederholt verwendete Werte werden berechnet

140...380...es werden 5 mal von je N Zahlen Mittelwert, Varianz etc. gebildet

150, 160, 170, 180...Initialisierung. M ist Mittelwert, S Varianz, U der kleinste Wert, O der größte Wert von N Zahlen. Der Index 1 ist für die uniforme Verteilung, Index 2 für die Gaußsche Verteilung.

190...340 ...Schleife für Mittelwert, Varianz und Extremwerte von N Zahlen

200...330...Schleife für uniforme (Index V=1) und normale (Index V=2) Verteilung

220...entweder Summe von K Zufallszahlen und Bildung einer Gaußschen Zufallszahl (260) oder sofort Untersuchung auf den kleinsten oder größten Wert

310...Summe der Zufallszahlen

320...Summe der Quadrate der Zufallszahlen

350,360...Varianz für uniforme und normale Verteilung wird gebildet (Mittelwert wird beim Ausdrucken in 370 gebildet)

```
RA#K       10:37   WIEN 02/06/71

1:      MITTEL            VARIANZ            MIN.              MAX.
2:  UNIF.   NORM.     UNIF.   NORM.     UNIF.   NORM.     UNIF.   NORM.
3:##.#### ##.####  ##.#### ##.####  ##.#### ##.####  ##.#### ##.####
10 PRINT "ERZEUGUNG NORMAL VERTEILTER ZUFALLSZAHLEN"
20 PRINT "VON UNIFORM VERTEILTEN ZUFALLSZAHLEN"
30 PRINT "PRUEFUNG VON MITTEL U. VARIANZ FUER BEIDE VERTEILUNGEN"
40 PRINT "ERSTE ZEILE ERWARTETE WERTE"
50 PRINT
60 N=100
70 K=12
80 K1=SQR(K/12)
90 K2=K/2
100 PRINTUSING 1
110 PRINTUSING 2
120 PRINTUSING 3,.5,0,.0833,1
130 PRINT
140 FOR I=1 TO 5
150 M(1)=M(2)=S(1)=S(2)=0
160 U(1)=U(2)=5E76
170 O(1)=O(2)=-5E76
180 R=RND(-1)
190 FOR L=1 TO N
200 FOR V=1 TO 2
210 R=RND(0)
220 ON V GOTO 270,230
230 FOR J=2 TO K
240 R=R+RND(0)
250 NEXT J
260 R=(R-K2)/K1
270 IF R<=O(V) THEN 290
280 O(V)=R
290 IF R>=U(V) THEN 310
```

```
300 U(V)=R
310 M(V)=M(V)+R
320 S(V)=S(V)+R*R
330 NEXT V
340 NEXT L
350 S(1)=(S(1)-M(1)*M(1)/N)/(N-1)
360 S(2)=(S(2)-M(2)*M(2)/N)/(N-1)
370 PRINTUSING 3,M(1)/N,M(2)/N,S(1),S(2),U(1),U(2),O(1),O(2)
380 NEXT I


 RA#K       10:38    WIEN 02/06/71

ERZEUGUNG NORMAL VERTEILTER ZUFALLSZAHLEN
VON UNIFORM VERTEILTEN ZUFALLSZAHLEN
PRUEFUNG VON MITTEL U. VARIANZ FUER BEIDE VERTEILUNGEN
ERSTE ZEILE ERWARTETE WERTE

     MITTEL              VARIANZ              MIN.                  MAX.
  UNIF.    NORM.      UNIF.    NORM.     UNIF.    NORM.      UNIF.    NORM.
  .5000    .0000      .0833   1.0000

  .4821   -.0809      .0860    .8756     .0045  -2.3922     .9868   2.5580
  .4945    .0027      .0924    .7424     .0050  -1.7413     .9983   2.7065
  .4636    .0925      .0910    .7041     .0061  -1.8370     .9827   2.2866
  .4181   -.0151      .0857    .8138     .0034  -2.5351     .9802   2.3072
  .5101    .0451      .1012    .9094     .0185  -1.6587     .9905   2.2582


USED     16.17 UNITS.
```

5.6 Zeichnen einer Kurve

Ein Problem, mit dem sich jeder Programmierer früher oder später beschäftigt, ist die "graphische Darstellung von Funktionen", d.h. das Zeichnen von Kurven mit dem Drucker.

Die TAB-Funktion, wie sie im BASIC des General Electric Betriebssystems Mark I/7 zur Verfügung steht, ist in diesem Zusammenhang von wesentlicher Bedeutung. Eine kurze Zusammenfassung der Eigenschaften der TAB-Funktion ist daher notwendig.

Grundsätzliche Form: PRINT TAB(Ausdruck);

Der Ausdruck wird mit den z.Z. der Aufrufung gültigen Werten der Variablen ausgewertet. Der ganzzahlige Teil des Ergebnisses wird, modulo 75, als Position zu der gerückt werden soll, interpretiert. Die Druckpositionen sind von links nach rechts von Ø bis 71 numeriert.

Zum Beispiel: Ergibt die Auswertung 7, wird auf die Druckposition 7 gerückt.
Ergibt die Auswertung 7.9999, wird auch auf die Druckposition 7 gerückt.
Ergibt die Auswertung 8Ø (oder 8Ø.7), wird auf die Druckposition 5 gerückt.

Steht der Schreibkopf in Druckposition N und wird die Anweisung PRINT TAB(M); gegeben und ist M<= N, wird die Anweisung übergangen (d.h. der Schreibkopf bleibt stehen, wo er ist).

Wegen der größeren Genauigkeit wird fast stets die unabhängige Variable vertikal und die abhängige Variable horizontal gezeichnet. Wegen der diskreten Zahl von Druckpositionen in einer Zeile (72 beim normalen TS–Terminal) muß über einen Skalierungs-, Verschiebungs- und Rundungs-Vorgang die Druckposition bestimmt werden. Die entsprechende Anweisung lautet z.B.

```
Y=INT(S*FNY(X)+Z+·5)
```

Darin ist S der Skalierungsfaktor, der der Anzahl der Druckpositionen für die Y-Einheit entspricht. Ist z.B. FNY(X)=SIN(X) und S=25, dann ist die Amplitude der Kurve gleich 25 Druckpositionen (d.h. einschließlich Maximum und Minimum gibt es 51 Druckpositionen). Für FNY(X)=5*SIN(X) und S=5 ist die Amplitude der Kurve ebenfalls 25 Druckpositionen. Ist FNY(X)=X/(1+X*X), eine Funktion, die bei x=1 ein Maximum von Ø.5 hat, und wird S=12Ø gemacht, dann liegt das Maximum der Kurve 6Ø Druckpositionen rechts von der "x-Achse".

Die Lage der x-Achse, d.h. der Punkte wo FNY(X)=Ø, wird durch die Größe von Z bestimmt. Für FNY(X)=SIN(X), wie oben, und Z=35, liegt die x-Achse in der Druckposition 35, d.h. ziemlich genau in der Mitte der Seite. Für FNY(X)=X/(1+X*X) und Z=5, ist die x-Achse in der 5ten Druckposition und die Kurve ist im Blatt zentriert.

Bleibt noch der Faktor Ø.5. Er dient zur Rundung und sorgt dafür, daß der Punkt in die dem tatsächlichen Wert nächstliegende Druckposition geschrieben wird. Ohne den Faktor Ø.5 würden z.B., Werte von Ø bis Ø.99999999 in die Stelle "Ø" und Werte zwischen –Ø.ØØØØØØØ1 und –Ø.99999999 in die Stelle "–1" gedruckt werden. Als Folge würde so eine zur x-Achse symmetrisch liegende Kurve (z.B. SIN(X) oben) unsymmetrisch gezeichnet werden.

Nun ist es ohne weiteres möglich, daß, gewollt oder ungewollt, die Kurve links und/oder rechts über den zur Verfügung stehenden Druckbereich hinausgeht. Dies kann zu Fehl-Positionierung führen, denn der Bereich der TAB-Funktion ist von Ø bis 74 (wegen modulo 75) und das Terminal hat tatsächlich nur 72 Druckpositionen. Von diesen 72 Positionen stehen die linken auch oft nicht zur Verfügung, weil in sie die x-Werte gedruckt werden. Es ist daher ratsam zwischen der Errechnung des y-Wertes (d.h. des Arguments der TAB-Funktion) und dem Drucken desselben abzufragen, ob

der Wert außerhalb der festgelegten Grenzen liegt. Z.B., wenn die Grenzen 1Ø und 7Ø (einschließlich) sind:

```
IF Y>7Ø THEN Zn
IF Y<1Ø THEN Zn
```

oder kürzer

```
IF ABS(Y-4Ø)>3Ø THEN Zn
```

wobei Sprung nach Zn das Ausdrucken überspringt.[1)]

Das eben Gesagte möge nun auf ein einfaches Beispiel angewendet werden. Es oll eine beliebige (durch DEF FNY(X) = definierte) Funktion gezeichnet werden. Wählbar (über READ/DATA) soll sein: Anfangs-, Endwert und Schrittgröße von x, sowie Skalierungsfaktor, Lage der x-Achse und Drucksymbol (eine $-Variable).

Besprechung:

1...Zeilenmaske für das Drucken des x-Wertes. Bei computergesteuertem Ausdrucken (d.h. mit PRINT) hätte man keine Kontrolle über das Druckformat.
1Ø...Definition der zu zeichnenden Funktion
2Ø...Einlesen von Anfangs-, Endwert und Schritt
4Ø...Einlesen von Skalierungsfaktor, Lage der x-Achse und Drucksymbol
6Ø...12Ø... Schleife für die verschiedenen x-Werte
7Ø...Ausdrucken der unabhängigen Variablen (und eines "Skalenstriches") und Stehenbleiben in der Zeile
8Ø...Errechnung der Druckposition
9Ø...Abfrage,ob Y innerhalb des zulässigen Druckbereiches (hier 11 bis 71): wenn außerhalb, dann neue Zeile (11Ø)
1ØØ...Drucken des Symbols in die errechnete Druckposition
11Ø...Vorrücken in neue Zeile

```
RA#S        10:49    WIEN 28/05/71

1 :##.###↑↑↑↑-
10 DEF FNY(X)=EXP(-X/2.5)*SIN(X)
20 READ A,E,D
30 DATA 0,12.8,.4
40 READ S,Z,P$
50 DATA 140,35,*
60 FOR X=A TO E STEP D
70 PRINTUSING 1, X,
80 Y=INT(S*FNY(X)+Z+.5)
90 IF ABS(Y-41)>30 THEN 110
100 PRINT TAB(Y);P$;
110 PRINT
120 NEXT X
130 GOTO 20
```

1) Wenn eine Überschreitung der Grenzen logisch unmöglich ist, kann diese Abfrage natürlich entfallen.

Soll eine andere Funktion gezeichnet werden, so sind die Anweisungen 1Ø, 3Ø, und 5Ø entsprechend zu ändern: 3Ø für Änderungen in der "x"-Skala; 5Ø für Änderungen in der "y"-Skala. In beiden Fällen können Bereich und Skalenfaktor gewählt werden.

```
RA#S        10:46    WIEN 28/05/71

 0.000E+00-                          *
 4.000E-01-
 8.000E-01-
 1.200E+00-
 1.600E+00-
 2.000E+00-
 2.400E+00-                                                                  *
 2.800E+00-                                              *
 3.200E+00-                        *
 3.600E+00-          *
 4.000E+00-    *
 4.400E+00-  *
 4.800E+00-     *
 5.200E+00-          *
 5.600E+00-                *
 6.000E+00-                     *
 6.400E+00-                           *
 6.800E+00-                               *
 7.200E+00-                                *
 7.600E+00-                                *
 8.000E+00-                                *
 8.400E+00-                              *
 8.800E+00-                            *
 9.200E+00-                           *
 9.600E+00-                         *
 1.000E+01-                         *
 1.040E+01-                        *
 1.080E+01-                        *
 1.120E+01-                        *
 1.160E+01-                         *
 1.200E+01-                         *
 1.240E+01-                          *
 1.280E+01-                          *

OUT OF DATA  IN 20


USED      2.50 UNITS.
```

Werte zwischen Ø,4 und 2,Ø (einschließlich) liegen rechts außerhalb des Druckbereiches.

5.7 Zeichnen einer Funktion unter Verwendung logarithmischer Teilung an Abszisse und Ordinate

Wenn sich der Bereich einer Variablen, sei es die abhängige oder die unabhängige, über mehrere Dekaden erstreckt, wird es notwendig, eine logarithmische Teilung zu verwenden.

Für die Abszisse kann die logarithmische Teilung grundsätzlich nach zwei Gesichtspunkten erfolgen:

Anfangswert (x_1) und Endwert (x_2) sind gegeben sowie

1. die gewünschte Zahl ($n + 1$) der Punkte pro Dekade
   oder
2. die Gesamtzahl ($m + 1$) der gewünschten Punkte zwischen Anfangs- und Endwert.

Im ersten Fall berechnet sich der Schrittfaktor aus $s=10^{(1/n)}$, im zweiten Fall aus

$s=(x_2/x_1)^{(1/m)}$.

Für die Ordinate bietet sich die direkte Verwendung einer logarithmischen Maßgröße an, z.B.der in der Elektronik weitverbreiteten Einheit Dezibel. Die Dezibel (Abkürzung dB) sind gegeben durch

$$dB = 20.\lg\frac{y}{y_0}$$

wobei y_0 eine Bezugsgröße ist.

Als Beispiel diene folgendes Problem: Die Funktion

$$y = x/(a^2 + x^2)^{1/2}$$

soll gezeichnet werden. Der Ordinatenbereich soll sich von 0 bis -30dB erstrecken, der Abszissenbereich soll wählbar (über READ/DATA) sein.

Besprechung:

10...Definierung der zu zeichnenden Funktion (bereits in dB; Bezugswert $y_0=1$)
50...Einlesen von Anfangswert (X) und Endwert (E) von X, Anzahl der Punkte pro Dekade (N) und eines Parameters (A)
70, 80...Skala
90...Berechnung des Schrittfaktors
100...Ausdrucken von X (und Stehenbleiben in der Zeile)
110...Skalierung (S=2 Druckpositionen pro Einheit, d.i. dB), Lage der Abszisse (im Beispiel in Druckposition 70) und Rundung
120...Abfrage,ob Y innerhalb des zugelassenen Bereiches (hier 10...70)
130...Ausdrucken des Y-Wertes
150...Berechnung des nächsten X-Wertes
160... Abfrage, ob X Endwert E erreicht hat

```
RA#A        9:34    WIEN 07/06/71

1:          ↑ . . . . : . . . . ↑ . . . . : . . . . ↑ . . . . : . . . .↑
2:##.##↑↑↑↑-
10 DEF FNY(X)=8.68588964*LOG(X/SQR(A↑2+X↑2))
20 PRINT "DARSTELLUNG EINER FUNKTION"
30 PRINT "MIT LOG. TEILUNG AUF ABSZISSE U. ORDINATE"
40 PRINT
50 READ X,E,N,A
60 DATA 5,2000,10,300
70 PRINT TAB(9);"-30";TAB(29);"-20";TAB(49);"-10";TAB(69);"0DB"
80 PRINTUSING 1
90 S=10↑(1/N)
100 PRINTUSING 2,X,
110 Y=INT(2*FNY(X)+70.5)
120 IF ABS(Y-40)>30 THEN 140
130 PRINT TAB(Y);"*"
140 PRINT
150 X=X*S
160 IF X<=E THEN 100
170 STOP



RA#A        10:26    WIEN 28/05/71

DARSTELLUNG EINER FUNKTION
MIT LOG. TEILUNG AUF ABSZISSE U. ORDINATE

          -30                  -20                  -10                  0DB
           ↑ . . . . : . . . . ↑ . . . . : . . . . ↑ . . . . : . . . . ↑
  5.00E+00-
  6.29E+00-
  7.92E+00-
  9.98E+00- *
  1.26E+01-    *
  1.58E+01-       *
  1.99E+01-          *
  2.51E+01-             *
  3.15E+01-                *
  3.97E+01-                   *
  5.00E+01-                      *
  6.29E+01-                         *
  7.92E+01-                            *
  9.98E+01-                               *
  1.26E+02-                                  *
  1.58E+02-                                     *
  1.99E+02-                                        *
  2.51E+02-                                           *
  3.15E+02-                                              *
  3.97E+02-                                                 *
  5.00E+02-                                                    *
  6.29E+02-                                                     *
  7.92E+02-                                                      *
  9.98E+02-                                                      *
  1.26E+03-                                                       *
  1.58E+03-                                                       *
  1.99E+03-                                                       *



USED      2.50 UNITS.
```

5.8 Zeichnen mehrerer Kurven

Etwas schwieriger wird das Kurvenzeichnen dann, wenn zwei oder mehr Kurven gleichzeitig gezeichnet werden sollen. Da der Schreibkopf schrittweise nur nach rechts (und nicht zurück nach links) rückt, müssen die Variablen erst ihrer Größe nach geordnet werden. Besetzen mehrere Variable die gleiche Druckposition, so muß entschieden werden, welche Drucksymbol Priorität hat.

Das Programm RAPL06 kann sechs Funktionen (mehr sind ohne weiteres möglich) der selben unabhängigen Variablen gleichzeitig drucken. Das Programm verwendet einen einfachen, aber sehr wirkungsvollen Algorithmus.

Es wird Y=INT(S*F(X)+Z+.5) berechnet und dann in die Zehntelstelle die Rangnummer der Funktion addiert. Dadurch gibt es keine "gleichen" Funktionswerte. Bei Gleichheit des ganzzahligen Teiles dominiert die Funktion mit der niedrigeren Rangnummer. Die Zehntelstelle etabliert die Priorität und identifiziert auch das zu verwendende Drucksymbol.

Besprechung:

1: und 2:...erlauben Eingabe beliebigen Textes
3:...ist die Skala
4:...ist die Maske für Drucken des x-Wertes
10...60...definieren bis maximal 6 Funktionen (erweiterbar)
100...liest Zahl der Funktionen (N), Anfangswert (A), Endwert (E) und Schrittgröße (D) der unabhängigen Variablen
120...140...liest Skalierungsfaktor S(I), Lage von Null Z(I) und Drucksymbol P$(I) für jede der N Funktionen ein
190...Beginn der Schleife für unabhängige Variable (hier T genannt)
200...bewirkt, daß nur die N rangniedrigsten Funktionen ausgewertet werden
210...260...die Funktionswerte kommen in ein Feld F(I)
270...290...den F(I) wird eine Druckposition zugeordnet und die Rangziffer addiert (+I/10)
300...Drucken des Wertes der unabhängigen Variablen
310...die Druckposition (P) wird initialisiert (mit einem zu kleinen Wert)
320...450...die Werte der N Funktionen werden für einen Wert der unabhängigen Variablen gedruckt (oder auch nicht)
330... Untersuchung, ob Funktionswert innerhalb des Druckbereiches (10...71) (Diese Abfrage muß auch Zehntel berücksichtigen, obwohl diese auf die Druckposition keinen Einfluß haben)
340...400...größenmäßiges Ordnen der N Funktionswerte
370...untersucht, ob betrachteter Funktionswert <10 und > M, wobei M der momentan kleinste unter den N Funktionswerten ist
410...verhindert mehrfaches Belegen einer Druckposition
430...rückt zur Druckposition und druckt das durch die Zehntel-Stelle codierte Drucksymbol

44Ø...noch nicht gedruckte Werte werden in Zellen bereits gedruckter geschrieben. D.h. die noch einzuordnenden Funktionswerte befinden sich immer bei den höheren Indexwerten von F(I), dadurch Abkürzung des Suchvorganges

46Ø...beginnt eine neue Zeile (43Ø bleibt in der Zeile stehen, in Erwartung eines weiteren Wertes)

```
RAPLO6      9:02    WIEN 26/05/71

DEMONSTRATIONS-BEISPIEL
O - X↑6    * - X↑5    + - X↑4    : - X↑3    = - X↑2    . - X↑1

           ↑....↑....↑....↑....↑....↑....↑....↑....↑....↑....↑....↑
-1.50E+00-.
-1.40E+00-  .
-1.30E+00-    .
-1.20E+00-      .                                                    =
-1.10E+00-   :    .                                              =    +
-1.00E+00-          *                                        O
-9.00E-01-            .  :  *                        O +  =
-8.00E-01-              .     :  *            O  +    =
-7.00E-01-                .      :   *    O  +    =
-6.00E-01-                  .       : *  O +   =
-5.00E-01-                    .      : *O+   =
-4.00E-01-                      .      :O+ =
-3.00E-01-                        .    :O =
-2.00E-01-                          .   O=
-1.00E-01-                            . O
-4.66E-09-                              O
 1.00E-01-                              O .
 2.00E-01-                              O=  .
 3.00E-01-                              O:=   .
 4.00E-01-                              O+ =    .
 5.00E-01-                              O*:  =    .
 6.00E-01-                               O*+:  =    .
 7.00E-01-                                O* + :  =   .
 8.00E-01-                                    O *+ :  =  .
 9.00E-01-                                        O*+ := .
 1.00E+00-                                                  O
 1.10E+00-                                                    . =  : +
 1.20E+00-                                                      .    =
 1.30E+00-                                                        .
 1.40E+00-                                                          .
 1.50E+00-                                                            .


USED      6.33 UNITS.
```

```
RAPLO6       8:59   WIEN 26/05/71

1:DEMONSTRATIONS-BEISPIEL
2:O - X↑6    * - X↑5    + - X↑4    : - X↑3    = - X↑2    . - X↑1
3:           ↑....↑....↑....↑....↑....↑....↑....↑....↑....↑....↑....↑....↑....↑...:
4:##.##↑↑↑↑-
10DEFFNA(X)=X↑6
20DEFFNB(X)=X↑5
30DEFFNC(X)=X↑4
40DEFFND(X)=X↑3
50DEFFNE(X)=X↑2
60DEFFNF(X)=X
100READN,A,E,D
110Z=N-1
120FORI=ØTOZ
130READS(I),Z(I),P$(I)
140NEXTI
150PRINTUSING1
160PRINTUSING2
170PRINT
180PRINTUSING3
190FORT=ATOESTEPD
200ONNGOTO260,250,240,230,220,210
210F(5)=FNF(T)
220F(4)=FNE(T)
230F(3)=FND(T)
240F(2)=FNC(T)
250F(1)=FNB(T)
260F(Ø)=FNA(T)
270FORI=ØTOZ
280F(I)=INT[S(I)*F(I)+Z(I)+.5]+I/10
290NEXTI
300PRINTUSING4,T;
310P=Ø
320FORJ=ØTOZ
330IFABS(F(J)-40.75)>30.8THEN450
340M=F(J)
350L=J
360FORK=J+1TOZ
370IFABS(2*F(K)-10-M)>M-10THEN400
380M=F(K)
390L=K
400NEXTK
410IFINT(M)=PTHEN440
420P=INT(M)
430PRINTTAB(P);P$[10*(M-P)+.5];
440F(L)=F(J)
450NEXTJ
460PRINT
470NEXTT
1000DATA6,-1.5,1.5,.1
1010DATA20,40,O,20,40,*,20,40,+,20,40,:,20,40,=,20,40,"."
```

5.9 Zeichnen in Polarkoordinaten

Zur Darstellung von Kreisen, Lissajous-Figuren und ähnlichen "verschlungenen" Kurven arbeitet man am besten in Polarkoordinaten. Am Drucker kann man natürlich nicht mit "Winkel-Strecke" arbeiten, sondern muß letzten Endes wieder "Schritte in der y-Richtung–Schritte in der x-Richtung" verwenden[1]. Es bietet sich daher, fast zwingend, eine zweidimensionale Darstellung der Form F(y,x) an.

Wegen der "digitalen" Arbeitsweise des Druckers (er kann nur g a n z e Druckpositionen nach rechts und g a n z e Zeilen nach unten rücken), fallen nur sehr wenige Punkte einer Kurve wirklich genau mit der Position eines Drucksymbols zusammen. Beim Zeichnen in kartesischen Koordinaten tritt dieses Problem nur quer über die Seite in Erscheinung, die Werte der Unabhängigen stimmen ja mit den Zeilen überein. Beim Zeichnen in Polarkoordinaten tritt auch eine Ungenauigkeit die Seite hinunter auf. Diese Ungenauigkeit ist besonders störend, denn der Vertikalabstand der Druckpositionen ist 5/3-mal größer als deren Horizontalabstand.

Werden die errechneten Punkte ohne Auswahl gedruckt, ergibt sich eine ungenaue (und unästhetische) "eckige" Kurve. Um die Kurve glatter und exakter zu machen, kann man die Punkte, die zu weit von der Druckpositionsmitte liegen, unterdrücken, muß aber dann gewisse Lücken im Kurvenverlauf akzeptieren. Um den Kurvenverlauf weiter zu glätten, kann man damit arbeiten, daß der Apostroph ' "oben" in einer Druckposition liegt, der Punkt . "unten" und Plus + in der Mitte.

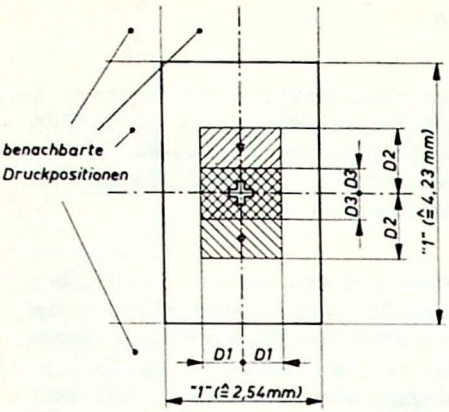
Für das Programm wurde eine Druckposition (ein Rechteck mit Höhe: Breite von 5:3), wie in der Abbildung gezeigt, unterteilt. Die in die unschraffierten Teile fallende Punkte werden nicht gedruckt, in die schraffierten Teile fallende Punkte werden, wie gezeigt, ausgedruckt. Die Wahl von D1, D2 und D3 ist etwas willkürlich, die im Programm verwendeten Werte (Ø,25–Ø,25–Ø,1) geben –auf Grund einer Versuchsserie– den im Durchschnitt glattesten und kontinuierlichsten Kurvenverlauf.

Für die Größe des Feldes wäre etwa 42 x 7Ø wünschenswert, weil es die ganze Seitenbreite ausnützen würde, jedoch ist dieses Feld für BASIC unzulässig groß. Verwendet wurde die Feldgröße 3Ø x 5Ø.

Besprechung:

1:...beliebiger Text (verwendet in 23Ø)
2, 3, 4, 5...Daten für READ-Anweisungen 11Ø, 14Ø, 15Ø, 18Ø
6...Reservierung von 31 Zeilen mal 51 Spalten

[1] Einfach über die Transformation: $y = r.\sin\varphi \quad x = r.\cos\varphi$



2Ø...9Ø...Definition von Funktionen. Unbedingt benötigt, d.h. im Programm direkt verwendet, wird FNV (S) und FNH (S). Die anderen Funktionen je nach Bedarf.
1ØØ...12Ø...Einlesen der Drucksymbole. Reihenfolge dient zur Etablierung einer Hierarchie (S$(I) dominiert über S$(I-1))
13Ø...Umrechnungsfaktor Grad-Radians
14Ø...Daten für "Größe" einer Druckposition (Siehe Bild)
15Ø...Einlesen der Lage des Ursprungs (YØ,XØ) und des Skalenfaktors horizontal (X1). (Die y-Richtung verläuft bei diesem Programm von oben nach unten, die x-Richtung von links nach rechts. Das ist unterschiedlich von den anderen Zeichenprogrammen) YØ...Zeilen, XØ...Druckpositionen; Ø...oberste Zeile, bzw. äußerste linke Druckposition, 3Ø...unterste Zeile, bzw. 5Ø...äußerste rechte Druckposition.
16Ø...Berechnung des zum horizontalen Skalenfaktor X1 gehörigen vertikalen Skalenfaktors Y1.
17Ø...Eintragung der Codeziffer für 0 (Ursprung)
18Ø...Einlesen von Beginnwinkel (A1), Endwinkel (A2) und Winkelschritt (A3) in Graden.
19Ø...22Ø...Initialisierung der Rekursionsformel für Sinus und Cosinus. Man beachte: S und C (und auch SØ und CØ) haben in diesem Programm die Bedeutung sin(A), cos(A) und werden als solche auch in den DEF FN. verwendet.
23Ø...Drucken der Überschriften
25Ø...43Ø...Hauptschleife
26Ø, 27Ø, 28Ø...Rekursionsformel für Sinus und Cosinus
29Ø...Berechnung des genauen y-Wertes
3ØØ...gerundeter, ganzzahliger y-Wert
31Ø...Verwerfung des Punktes, wenn außerhalb der Zeilen Ø bis 3Ø
32Ø...Differenz zwischen ganzzahligem und genauem Wert

330...Verwerfung des Punktes, wenn außerhalb der Druckzone ± D2
340...Berechnung des genauen x-Wertes
350...gerundeter, ganzzahliger x-Wert
360...Verwerfung des Punktes, wenn außerhalb der Druckzone ± D1
370...Verwerfung des Punktes, wenn außerhalb der Druckposition 0 bis 50
380, 390, 400...Zuordnung einer Codeziffer entsprechend der Lage des Punktes.
3...wenn innerhalb ± D3-Zone; 2...wenn unterhalb D3; 1...wenn oberhalb D3.
(Codeziffer 0 bedeutet Leerstelle)
410...Verwerfung des Punktes, wenn neue Codeziffer kleiner als bereits eingetragene
420...Eintragung der Codeziffer
440...480...Aufsuchung der niedrigsten (d.h. obersten) Zeile, in der sich ein "Punkt" befindet. Durch Verwendung des gefundenen Wertes in 490 Unterdrückung von "führenden" Leerzeilen.
490...550...Ausdrucken der Kurve

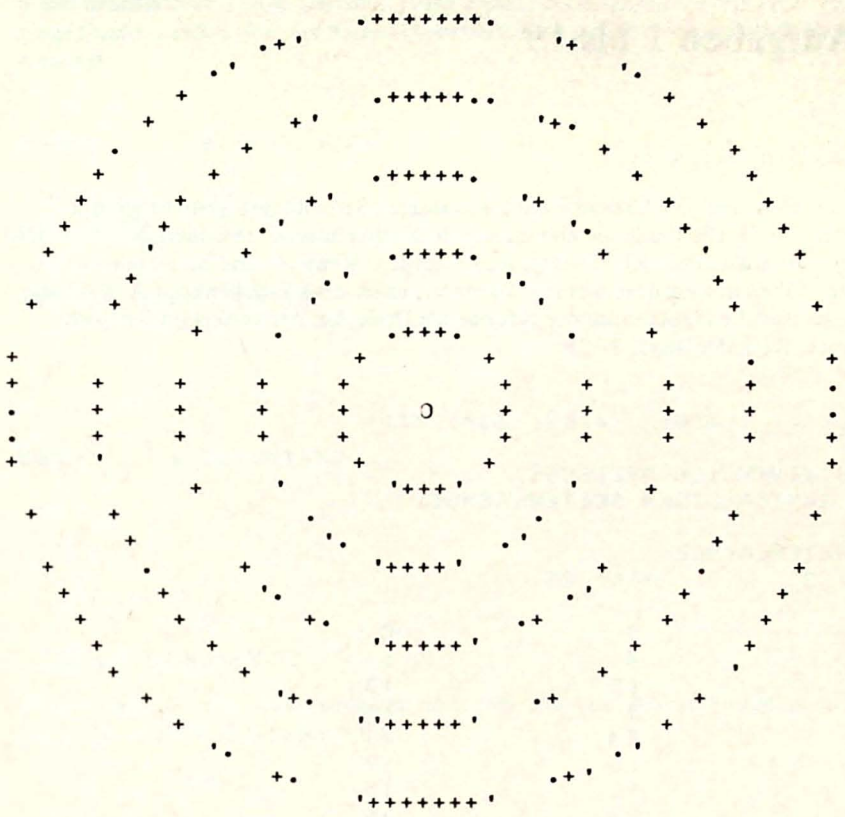
Das Programmbeispiel zeigt ein Antennenstrahlungsdiagramm. Als weitere Beispiele sind fünf konzentrische Kreise sowie die olympischen Ringe gezeichnet. Als Übungsaufgabe ändere man die DEF-Anweisungen und die DATA-Anweisungen, sodaß die konzentrischen Kreise bzw. die olympischen Ringe gezeichnet werden.

```
RAPOM        8:48    WIEN 03/09/71

1:VERT. HALBWELLENDIPOL UEBER ERDE.  H=4[M]  WL=2[M]
2DATA',".",+,)
3DATA.25,.25,.1
4DATA30,0,50
5DATA0,89.8,.1
6DIMF(30,50)
20DEFFNE(S)=COS(1.5707963*S)/C
30DEFFNG(S)=SIN(12.5663706*S)
40DEFFNA(S)=ABS(FNE(S)*FNG(S))
50DEFFNV(S)=FNA(S)*S
60DEFFNH(S)=FNA(S)*C
100FORI=1TO4
110READS$(I)
120NEXTI
130R=1.7453292E-2
140READD1,D2,D3
150READY0,X0,X1
160Y1=X1*.6
170F(Y0,X0)=4
180READA1,A2,A3
190S=SIN(R*(A1-A3))
200C=COS(R*(A1-A3))
210S0=SIN(R*A3)
220C0=COS(R*A3)
230PRINTUSING1
240PRINT
250FORA=A1TOA2STEPA3
260T=C*C0-S*S0
```

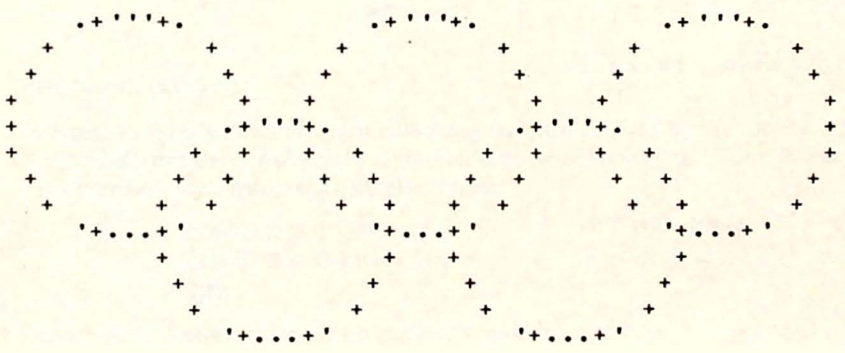
```
270S=S*C0+C*S0
280C=T
290Y=Y0-Y1*FNV(S)
300V=INT(Y+.5)
310IFABS(V-15)>15THEN430
320D=V-Y
330IFABS(D)>D2THEN430
340X=X1*FNH(S)+X0
350H=INT(X+.5)
360IFABS(X-H)>D1THEN430
370IFABS(H-25)>25THEN430
380T=3
390IFABS(D)<=D3THEN410
400T=INT(2-.5*SGN(D))
410IFT<=F(V,H)THEN430
420F(V,H)=T
430NEXTA
440FORA=0TO30
450FORH=0TO50
460IFF(A,H)<>0THEN490
470NEXTH
480NEXTA
490FORV=ATO30
500FORH=0TO50
510IFF(V,H)=0THEN530
520PRINTTAB(H);S$[F(V,H)];
530NEXTH
540PRINT
550NEXTV
```

```
RAPOM        8:18   WIEN 02/07/71

VERT. HALBWELLENDIPOL UEBER ERDE.  H=4[M]  WL=2[M]

                         .+'+
                      .+   +
           .        +     +
         ' +      '     +           ..+'''
       +        +     +        .+'      +'
      +   +   +     +      +'        .'
     +   +  '     +     '         .'
    +   +  '    '   '         .'
   +   '+    . .+         .'
  +  ..     '.'      .+'         .. ++'''''   '''+.
    ++   +      .+'  ..++'                      .+
 + + .+                                   ..+'
..   .+'                      .  +''
0++       +'''''

USED    21.67 UNITS.
```

RAPOM 8:22 WIEN 02/07/71
KONZENTRISCHE KREISE



RAPOM 9:04 WIEN 03/09/71



6. Aufgaben 1 bis 19

Aufgabe 1 (Lösung S. 71)

Es sollen rechtwinklige Dreiecke mit ganzzahligen Seitenlängen gesucht werden, (wie z.B. 3,4,5). Die minimale und maximale Kathetenlänge, zwischen denen gesucht werden soll, sollen über READ eingegeben werden. Minimale und maximale Kathetenlänge sollen ausgedruckt werden. Darunter, unter dem Tabellenkopf A B C, die Katheten und die Hypotenuse der gefundenen Dreiecke. Als konkretes Beispiel: min.-max. Kathetenlänge 3–25.

```
RA#C       14:01    WIEN  03/11/71

RECHTWINKELIGE DREIECKE
MIT GANZZAHLIGEN SEITENLAENGEN

KATHETENLAENGE
MIN: 3          MAX: 25

 A              B              C
 3              4              5
 5              12             13
 6              8              10
 7              24             25
 8              15             17
 9              12             15
 10             24             26
 12             16             20
 15             20             25
 18             24             30
 20             21             29

OUT OF DATA  IN 40



USED      3.00 UNITS.
```

Aufgabe 2 (Lösung S. 72)

Es soll eine Zahl A in Fixkomma-Format (dreistellig, zwei Dezimalstellen) und unmittelbar anschließend (ohne Leerstelle) eine Zahl B in Ganzzahlen-Format (zweistellig) ausgedruckt werden. Für das Beispiel verwende man positive und negative Werte für A und B.

```
RA#01       12:06    WIEN  05/10/71

-1.23-45
 6.78 90
 35.71-1317

OUT OF DATA  IN 10



USED       1.50 UNITS.
```

Aufgabe 3 (Lösung S. 72)

Man schreibe e i n e Anweisung, die nach 1ØØ, 2ØØ oder 3ØØ verzweigt, je nach dem, ob die Variable V $<$, =, oder $>$ Ø ist.

Aufgabe 4 (Lösung S. 74)

In einem Programm befindet sich eine Anweisungsfolge zur Abfrage, ob die Variable X innerhalb des Bereiches von U bis 0 (beide einschließlich) liegt. Wenn X außerhalb des Bereiches liegt, wird nach 5ØØ gesprungen.

```
2ØØ IF X > 0 THEN 5ØØ
21Ø IF X < U THEN 5ØØ
22Ø...
```

Diese beiden Anweisungen sollen in einer vereinigt werden.

Aufgabe 5 (Lösung S. 74)

Beim Ausdrucken einer n-zeiligen Tabelle soll in m Zeilen gruppiert werden. Nach je m Zeilen soll eine Leerzeile folgen und nach k Gruppen von m Zeilen zwei Leerzeilen. Für m=2 und k=2, zum Beispiel.

```
XXXXX
XXXXX
- - -
XXXXX
XXXXX
- - -

- - -
XXXXX
XXXXX
- - -
```

Für das Beispiel n=14, m=3, k=2.

```
RA#U        15:55    WIEN   06/10/71

 1
 2
 3

 4
 5
 6


 7
 8
 9

 10
 11
 12


 13
 14



USED        1.50 UNITS.
```

Aufgabe 6 (Lösung S. 75)

Es sind 1Ø ganzzahlige Zufallszahlen zwischen Ø und 1ØØ zu erzeugen. Diese Zahlen sind, eine pro Zeile, auszudrucken. Hinter der Zahl soll stehen: UNGERADE, bzw. GERADE, je nachdem, was zutrifft. U von UNGERADE, bzw. G von GERADE, soll in der 7ten Druckposition stehen (die äußerste linke Druckposition ist die Øte)

```
RA#O        10:27    WIEN 02/06/71

 93      UNGERADE
 1       UNGERADE
 19      UNGERADE
 42      GERADE
 81      UNGERADE
 6Ø      GERADE
 14      GERADE
 91      UNGERADE
 52      GERADE
 7Ø      GERADE

USED       1.5Ø UNITS.
```

Aufgabe 7 (Lösung S. 75)

In einem Programm wird eine Reihe von Tabellen berechnet. Diese Tabellen sollen die Überschrift TABELLE A, TABELLE B etc., erhalten. Die Tabellen werden in einer FOR-Schleife mit Schleifenvariabler I=1...15 berechnet. Man schreibe die Programmteile, die zum Ausdrucken der gewünschten Titel notwendig sind. Für das Problembeispiel: drucke TABELLE A, in die nächste Zeile TABELLE B, usw.

```
RA#M         9:12    WIEN Ø7/Ø6/71

TABELLE A
TABELLE B
TABELLE C
TABELLE D
TABELLE E
TABELLE F
TABELLE G
TABELLE H
TABELLE I
TABELLE J
TABELLE K
TABELLE L
TABELLE M
TABELLE N
TABELLE O

USED       1.5Ø UNITS.
```

Aufgabe 8 (Lösung S. 76)

Angenommen ein Programm ordnet N (maximal 5Ø) Zahlen in steigender oder fallender Folge. Für steigend wird eine Variable J=1 gesetzt, für fallend J=2. N und J werden über READ N, J eingelesen. Es soll ein Tabellenkopf folgender Form geschrieben werden:

ORDNEN VON (Wert von N) ZAHLEN IN (steigender, bzw. fallender) FOLGE

Man schreibe die Programmteile, die zum gewünschten Titel führen.

```
RA#W       10:17    WIEN 31/Ø8/71

ORDNEN VON    9 ZAHLEN IN STEIGENDER FOLGE

ORDNEN VON    9 ZAHLEN IN  FALLENDER FOLGE

ORDNEN VON   2Ø ZAHLEN IN STEIGENDER FOLGE

ORDNEN VON   1ØØ ZAHLEN IN  FALLENDER FOLGE

OUT OF DATA  IN 3Ø



USED       1.5Ø UNITS.
```

Aufgabe 9 (Lösung S. 76)

In einem Programm werden sechs Zahlen eingelesen, als Ergebnis gibt es drei Zahlen. Die Werte aller neun Zahlen können mehrere Dekaden überstreichen (d.h. sie können Werte zwischen, z.B. $1Ø^{6}$ und $1Ø^{-6}$ haben). Die Eingabegrößen mögen X, Y, Z, A1, A2, A3 heißen;die Ausgabegrößen L, PA, PB. Die Größen sollen in der Form

X=..... Y=..... Z=.....

immer drei pro Zeile, ausgedruckt werden. Die geforderte Genauigkeit ist vier Stellen.

Die drei Zeilen sollen mit drei PRINTUSING-Anweisungen, aber nur einer Zeilenmaske, geschrieben werden.

Man schreibe die Teile des Programms, die zur gewünschten Ausgabe führen. Die neun Zahlen sollen über READ eingelesen werden. Für den Testlauf sollen Zahlen, die den ganzen Zahlenbereich des Computers überstreichen, gewählt werden (einschließlich Ø).

```
RA#X        10:14    WIEN 31/08/71

  X= 1.234E+00    Y= 5.678E+09    Z= 1.230E-06
A1= 5.000E+00  A2=-6.000E+00  A3= 7.890E+02
 L= 0.000E+00  PA= 1.200E+35  PB= 5.000E-67


OUT OF DATA  IN 10



USED       1.50 UNITS.
```

Aufgabe 10 (Lösung S. 76)

Man drucke ein Schachbrettmuster. Die Symbole für schwarz und weiß sollen über READ/DATA wählbar sein. Für den Test, * für schwarz, Blank für weiß (Maßstabinformation: Auf Terminals Teletype ASR 33 sind 5 horizontale Druckpositionen längenmäßig gleich 3 Zeilen, d.h. 5 * nebeneinander und drei dieser 5er Gruppen übereinander ergeben ein quadratisches Gebilde). Zum Prüfen des Programms, enthalte das Feld 4x4 Quadrate. (Dies ist ein schönes FOR-Schleifen-Beispiel)

```
RA#P        13:23    WIEN 02/06/71

*****       *****
*****       *****
*****       *****
      *****       *****
      *****       *****
      *****       *****
*****       *****
*****       *****
*****       *****
      *****       *****
      *****       *****
      *****       *****



USED       2.17 UNITS.
```

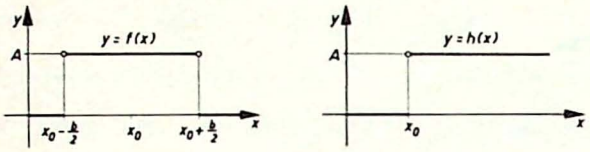
Aufgabe 11 (Lösung S. 77)

Wenn eine Funktion nur zwischen zwei Werten des Arguments definiert ist (und außerhalb Null ist), dann kann dies in einer Gleichung formuliert werden:

$$f(x)=A.\text{rect}\left(\frac{x-x_0}{b}\right).\ g(x)$$

Die rect-Funktion kann auch aus der Superposition zweier Sprungfunktionen (eine positiv gehend, eine negativ gehend) entstanden, gedacht werden.

rect-Funktion und Sprungfunktion sind in der folgenden Abbildung definiert.



$f(x)=A.\text{rect}\ \frac{x-x_0}{b}$ rect-Funktion

$h(x)=A.u(x-x_0)$ Sprung-Funktion

Anwendung finden diese Funktionen z.B. in der Elektronik (Impulstechnik). Im besonderen wird ein idealisierter Radar-Puls beschrieben durch

$$f(t)=A.\text{rect}\left(\frac{t-t_0}{T}\right).\sin(2\pi ft)$$

Es ist erwünscht diese Funktion in einer einzeiligen Anweisung zu schreiben, z.B. als DEF FNP (T)= ... .

Unter Verwendung des Programmes RA#S, zeichne man zwei volle Perioden einer Sinusfunktion (siehe Programm-Ausgabe).

```
RA#S        9:47    WIEN 02/06/71

 0.000E+00-                        *
 1.000E+00-                        *
 2.000E+00-                        *
 3.000E+00-                        *
 4.000E+00-                              *
 5.000E+00-                                    *
 6.000E+00-                                        *
 7.000E+00-                                          *
 8.000E+00-                                        *
 9.000E+00-                                    *
 1.000E+01-                              *
 1.100E+01-                        *
 1.200E+01-                  *
 1.300E+01-            *
 1.400E+01-         *
 1.500E+01-        *
 1.600E+01-         *
 1.700E+01-            *
 1.800E+01-                  *
 1.900E+01-                        *
 2.000E+01-                              *
 2.100E+01-                                    *
 2.200E+01-                                        *
 2.300E+01-                                          *
 2.400E+01-                                        *
 2.500E+01-                                    *
 2.600E+01-                              *
 2.700E+01-                        *
 2.800E+01-                  *
 2.900E+01-            *
 3.000E+01-         *
 3.100E+01-        *
 3.200E+01-         *
 3.300E+01-            *
 3.400E+01-                  *
 3.500E+01-                        *
 3.600E+01-                        *
 3.700E+01-                        *
 3.800E+01-                        *
 3.900E+01-                       @*
 4.000E+01-                        *

OUT OF DATA  IN 20

USED      2.83 UNITS.
```

Das Zeichen @ in der vorletzten Zeile ist durch eine Leitungsstörung entstanden (Durch einen zusätzlichen Puls wurde aus Blank @)

Aufgabe 12 (Lösung S. 78)

Man wandle ganze positive Zahlen (zwischen 1 und 4001 z.B.) in römische Zahlen um. Schreibregeln für römische Zahlen M=1000, D=500, C=100, L= 50, X=10, V=5, I=1. Maximale Aufeinanderfolge gleicher Zeichen: M beliebig (bzw. 4), D...1, C...3, L...1, X...3, V...1, I...3. Vorgesetzt darf nur die nächstniedrigere 10er-Potenz werden (z.B. CM, CD, XC, XL etc. Nicht jedoch IC, LD, VL etc.)

```
RA#I        10:34    WIEN 02/06/71

UMWANDLUNG ARAB. IN ROEM. ZAHLEN

  3888            MMMDCCCLXXXVIII
  276             CCLXXVI
  2555            MMDLV
  444             CDXLIV
  999             CMXCIX
  777             DCCLXXVII
  331             CCCXXXI
  95              XCV
 -5               NUR 1...4001
  4002            NUR 1...4001
  4001            MMMMI
  1971            MCMLXXI

OUT OF DATA  IN 90



USED       2.50 UNITS.
```

Aufgabe 13 (Lösung S. 79)

Es sei eine Tabelle für x, x^2, x^3, $x^{1/2}$, $x^{1/3}$, $\ln x$, $\lg x$, e^x, $1/x$ zu erzeugen. Die Tabelle soll einen passenden Kopf haben (z.B. so wie oben aufgeschrieben) und x soll von 1, 2, 3...10 gehen. In allen Spalten, in denen Dezimalstellen vorkommen, sollen 5 Dezimalstellen ausgedruckt werden.

Aufgabe 14 (Lösung S. 80)

Für die Transformation von kartesischen Koordinaten in Polarkoordinaten, wie sie z.B. bei der Arbeit mit komplexen Zahlen auftreten kann, gilt:

$$z = x + j \cdot y \longrightarrow \text{Betrag } z = \sqrt{x^2+y^2} \qquad \text{Winkel } z = \arctan\frac{y}{x}$$

Nun liefert die eingebaute Funktion ATN( ) nur den Hauptwert, d.h. Winkel zwischen $+\frac{\pi}{2}$ und $-\frac{\pi}{2}$. Daher sind Punkte im 2ten und 3ten Quadranten nicht direkt mit ATN( ) berechenbar. Darüber hinaus kann bei Anwendung von ATN(Y/X) die Fehlermeldung "DIVISION BY ZERO" auftreten.

Es ist ein Programm zu schreiben, welches möglichst wirtschaftlich, d.h. mit möglichst wenig Zeichen, den Bereich von ATN( ) auf $+\pi$ bis $-\pi$ ausdehnt und gleichzeitig die Fälle mit Realteil gleich Null handhaben kann.

```
RA#F        12:46    WIEN 26/05/71

TRANSFORMATION VON
KART. KOORD. IN POLAR KOORD.

X= 0                Y= 0
R= 0                P= 0

X= 0                Y= 2
R= 2                P= 90

X= 0                Y=-3
R= 3                P=-90.

X= 4                Y= 0
R= 4                P= 0

X=-5                Y= 0
R= 5                P= 180

X=-.75471           Y= .656059
R= 1                P= 139.

X=-.325568          Y=-.945519
R= 1                P=-109.


OUT OF DATA  IN 40



USED      1.83 UNITS.
```

Aufgabe 15 (Lösung S. 80)

Es soll ein in Radians gegebener Winkel in Grad, Minuten und Sekunden (gerundet auf Zehntel-Sekunden) umgewandelt werden. Grad, Minuten und Sekunden sollen gespeichert sein, d.h. kein Rechnen/Runden soll in der PRINT-Anweisung erfolgen. Als Testwerte benütze man u.a. $\pi/2$, π, 2π.

```
RA#LA      10:51    WIEN 28/05/71

   1.5707963000[RAD] =    90[°]   0[']    .0["]
   3.1415926000[RAD] =   180[°]   0[']    .0["]
   6.2831853000[RAD] =   360[°]   0[']    .0["]
    .0174532920[RAD] =     1[°]   0[']    .0["]
    .0002908882[RAD] =     0[°]   1[']    .0["]
    .0000048481[RAD] =     0[°]   0[']   1.0["]
   1.0000000000[RAD] =    57[°]  17[']  44.8["]
 174.5329200000[RAD] =  9999[°]  59[']  58.8["]
 174.5329200000[RAD] =  9999[°]  59[']  59.9["]
1745.3292000000[RAD] = 99999[°]  59[']  58.0["]

OUT OF DATA  IN 10


USED      2.33 UNITS.
```

Aufgabe 16 (Lösung S. 81)

Man entwickle einen Algorithmus zum Ausdrucken des Pascalschen Dreiecks. Am linken Rande sollen die Zahlen von 0 bis 10 stehen, entsprechend der Ordnung, und rechts davon, in Form eines gleichschenkligen Dreiecks, die Binominalkoeffizienten.

```
RA#J       12:57    WIEN 26/05/71

PASCAL' DREIECK

 0                                  1
 1                               1     1
 2                            1     2     1
 3                         1     3     3     1
 4                      1     4     6     4     1
 5                   1     5    10    10     5     1
 6                1     6    15    20    15     6     1
 7             1     7    21    35    35    21     7     1
 8          1     8    28    56    70    56    28     8     1
 9       1     9    36    84   126   126    84    36     9     1
10    1    10    45   120   210   252   210   120    45    10     1


USED      2.50 UNITS.
```

Aufgabe 17 (Lösung S. 82)

Die Fibonacci-Folge ist eine Zahlenreihe, bei der jedes Glied gleich der Summe der beiden vorhergehenden ist. Zum Beispiel

1 2 3 5 8 13 21 34 55 .....

Unabhängig von den Anfangsgliedern, wenn nur genügend Glieder gebildet werden, strebt das Verhältnis zweier aufeinanderfolgender Glieder dem Wert 1,618Ø3398..., dem "goldenen Verhältnis" zu (der Reziprokwert des goldenen Verhältnis ist Ø,618Ø3398...).

Verallgemeinert man und läßt jedes Glied gleich der Summe von n vorhergehenden sein, dann wird das Verhältnis, für n gegen ∞, gleich 2.

Es ist ein Programm zu schreiben, welches den Grenzwert, dem das Verhältnis a_n/a_{n-1} zustrebt, bestimmt.

Eingabe, über READ/DATA, n, die Zahl der zu summierenden Glieder zur Bildung des nächsten, und die ersten n Glieder ($2 \leq n \leq 11$).

Ausgabe: DIE ERSTEN ... GLIEDER und darunter die ersten n Glieder; DIE LETZTEN ... GLIEDER und darunter die letzten n Glieder; A(N)/A(N−1)= und nach dem Gleichheitszeichen der Wert des Verhältnisses.

Die Summierung soll abgebrochen werden, wenn entweder das Verhältnis bis zur 5ten Dezimalstelle gleich mit dem vorhergehenden Verhältnis ist oder wenn bereits 1ØØ Glieder gebildet wurden und das Verhältnis noch immer nicht die gewünschte Genauigkeit erreicht hat. Wenn der letztere Fall eintritt, ist eine entsprechende Meldung auszudrucken.

Nach Beendigung einer Rechnung soll wieder zum Einlesen neuer Daten gesprungen werden.

```
RA#V       14:42    WIEN  24/09/71

VERALLGEMEINERTE FIBONACCI-FOLGE

DIE  ERSTEN 2 GLIEDER
 Ø              1

DIE LETZTEN 2 GLIEDER
 1597           2584

A(N)/A(N-1)= 1.61803


DIE  ERSTEN 5 GLIEDER
-3              1Ø              54             -33             14

DIE LETZTEN 5 GLIEDER
 129782         255143          501608         986134          1938685
```

```
A(N)/A(N-1)= 1.96594
DIE  ERSTEN 11   GLIEDER
 0              1              2              3              4
 5              6              7              8              9
 10

DIE LETZTEN 11   GLIEDER
 436            869            1734           3463           6920
 13833          27658          55307          110604         221153
 442196

A(N)/A(N-1)= 1.9995

OUT OF DATA  IN 50

USED     3.67 UNITS.
```

Aufgabe 18 (Lösung S. 83)

Die Zahl π, das Verhältnis von Umfang zu Durchmesser eines Kreises, soll über Umfang eines $2n$-Eckes, $n \to \infty$, angenähert werden. Der Näherungswert für π soll für ein 8,16,32,....65536-Eck (d.i. $2^3 \ldots 2^{16}$) ausgedruckt werden. Anordnung der Ausgabe:

```
    8      Wert für π
   16      Wert für π
    .
    .
    .
    .
65536      Wert für π
```

(π selber darf im Programm natürlich nicht verwendet werden)

```
RA#B1        9:12     WIEN 19/05/71

PI UEBER 2N-ECK

       8      3.0614675
      16      3.1214451
      32      3.1365485
      64      3.1403311
     128      3.1412772
     256      3.1415138
     512      3.1415729
    1024      3.1415877
    2048      3.1415914
    4096      3.1415923
    8192      3.1415925
   16384      3.1415926
   32768      3.1415926
   65536      3.1415926

USED      2.00 UNITS.
```

Aufgabe 19 (Lösung S. 84)

Differenzentafeln haben in der numerischen Mathematik, z.B. bei Interpolation, eine Bedeutung. In der Programmierung stellen sie eine gute Übung für das Arbeiten mit indizierten Variablen, sowie für die Handhabung des Ausdruckformats dar.

Zur Erklärung einer Differenzentafel diene als Beispiel die Differenzentafel für $y=x^2$, x=Ø...1Ø

	Y		D1		D2		D3
y_0	Ø						
y_1	1	d_{11}	1				
y_2	4	d_{21}	3	d_{22}	2		
y_3	9	d_{31}	5	d_{32}	2	d_{33}	Ø
y_4	16	d_{41}	7	d_{42}	2	d_{43}	Ø
y_5	25	d_{51}	9	d_{52}	2	d_{53}	Ø
.							
.							
.							
.							
$y_{1Ø}$	1ØØ	$d_{1Ø1}$	19	$d_{1Ø2}$	2	$d_{1Ø3}$	Ø

$d_{11}=y_1-y_0$ $d_{21}=y_2-y_1$ .....

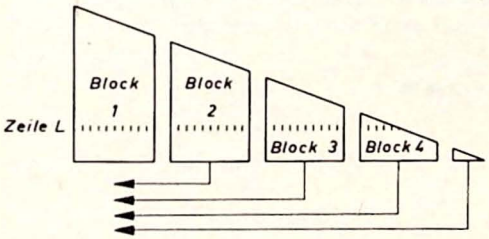
$d_{22}=d_{21}-d_{11}$ $d_{32}=d_{31}-d_{21}$ .....

$d_{33}=d_{32}-d_{22}$ $d_{43}=d_{42}-d_{32}$ .....

Da der dritte Differentialquotient von x^2 gleich Ø ist, endet die Differenzentafel bei *D*3 (d.h. von da an sind alle d=Ø). Sind von einer Funktion beliebig viele Differentialquotienten ungleich Null, dann endet die Tabelle bei $D(N-1)$, wobei N die Anzahl der gegebenen Funktionswerte ist.

Es soll ein Programm geschrieben werden, welches eine Differenzentafel für eine beliebige Funktion errechnet und dann in einem zweidimensionalen Feld speichert (Zur Verwendung in einem hier nicht vorhandenen Hauptprogramm).

Die Tafel soll auch ausgedruckt werden. Nun ist aber in einer Zeile nur eine begrenzte Anzahl von Zahlen unterzubringen. Der verbleibende Teil des Tabellen-Dreiecks muß dann weiter unten fortgesetzt werden, etwa derart:



Block 2 unter Block 1, Block 3 unter Block 2, etc. Für jeden Block muß links eine y-Spalte geschrieben werden.

Als konkretes Beispiel ist die Differenzentabelle von $y=\sin\left(\frac{\pi}{10}\cdot n\right)$, $n=0...20$ zu bilden. Differenzen auf 5 Dezimalstellen (d.h. zum Beispiel Maske ##.#####)

```
RA#H       12:40    WIEN 05/06/71

DIFFERENZEN-TAFEL

     Y      D 1     D 2     D 3     D 4     D 5     D 6     D 7     D 8
  .00000
  .30902  .30902
  .58779  .27877 -.03025
  .80902  .22123 -.05754 -.02729
  .95106  .14204 -.07919 -.02166  .00563
 1.00000  .04894 -.09310 -.01390  .00775  .00212
  .95106 -.04894 -.09789 -.00479  .00911  .00136 -.00076
  .80902 -.14204 -.09310  .00479  .00958  .00047 -.00089 -.00013
  .58779 -.22123 -.07919  .01390  .00911 -.00047 -.00094 -.00005  .00009
  .30902 -.27877 -.05754  .02166  .00775 -.00136 -.00089  .00005  .00009
  .00000 -.30902 -.03025  .02729  .00563 -.00212 -.00076  .00013  .00009
 -.30902 -.30902  .00000  .03025  .00296 -.00267 -.00055  .00021  .00007
 -.58779 -.27877  .03025  .03025 -.00000 -.00296 -.00029  .00026  .00005
 -.80902 -.22123  .05754  .02729 -.00296 -.00296  .00000  .00029  .00003
 -.95106 -.14204  .07919  .02166 -.00563 -.00267  .00029  .00029 -.00000
-1.00000 -.04894  .09310  .01390 -.00775 -.00212  .00055  .00026 -.00003
 -.95106  .04894  .09789  .00479 -.00911 -.00136  .00076  .00021 -.00005
 -.80902  .14204  .09310 -.00479 -.00958 -.00047  .00089  .00013 -.00007
 -.58779  .22123  .07919 -.01390 -.00911  .00047  .00094  .00005 -.00009
 -.30902  .27877  .05754 -.02166 -.00775  .00136  .00089 -.00005 -.00009
 -.00000  .30902  .03025 -.02729 -.00563  .00212  .00076 -.00013 -.00009

     Y      D 9     D 10    D 11    D 12    D 13    D 14    D 15    D 16
  .58779
  .30902  .00000
  .00000 -.00000 -.00001
 -.30902 -.00001 -.00001 -.00001
 -.58779 -.00002 -.00001  .00001  .00001
 -.80902 -.00003 -.00001 -.00000 -.00001 -.00002
 -.95106 -.00003 -.00000  .00000  .00001  .00001  .00003
-1.00000 -.00003  .00000  .00000 -.00000 -.00001 -.00002 -.00005
 -.95106 -.00003  .00000  .00000 -.00000 -.00000  .00000  .00002  .00007
 -.80902 -.00002  .00001  .00001  .00001  .00001  .00001  .00001 -.00001
 -.58779 -.00001  .00000 -.00000 -.00001 -.00002 -.00003 -.00004 -.00005
 -.30902 -.00000  .00001  .00001  .00001  .00002  .00004  .00007  .00011
 -.00000  .00000  .00001 -.00001 -.00002 -.00003 -.00005 -.00010 -.00017

     Y      D 17    D 18    D 19    D 20    D 21    D 22    D 23    D 24
 -.95106
 -.80902 -.00008
 -.58779 -.00004  .00005
 -.30902  .00016  .00020  .00015
 -.00000 -.00028 -.00044 -.00063 -.00078


USED       7.50 UNITS.
```

Lösungen zu den Aufgaben

Aufgabe 1 (Angaben S. 56)

Das Problem wird mit einer Doppelschleife gelöst. Dadurch wird ein Duplizieren von Dreiecken vermieden (wie 8, 15, 17; 15, 8, 17; 17, 8, 15). In den Schleifen 1ØØ und 11Ø werden A und B in Schritten von 1 erhöht. 12Ø berechnet die Hypotenuse, und 13Ø prüft, ob das Ergebnis ganzzahlig ist. Wenn ja, wird ausgedruckt (14Ø); wenn nein, in der Schleife unmittelbar fortgesetzt.

```
RA#C        14:ØØ    WIEN  Ø3/11/71

1Ø PRINT "RECHTWINKELIGE DREIECKE"
2Ø PRINT "MIT GANZZAHLIGEN SEITENLAENGEN"
3Ø PRINT
4Ø READ A1,A2
5Ø DATA 3,25
6Ø PRINT "KATHETENLAENGE"
7Ø PRINT "MIN:";A1,"MAX:";A2
8Ø PRINT
9Ø PRINT " A"," B"," C"
1ØØ FOR A=A1 TO A2
11Ø FOR B=A TO A2
12Ø C=SQR(A↑2+B↑2)
13Ø IF INT(C)<>C THEN 15Ø
14Ø PRINT A,B,C
15Ø NEXT B
16Ø NEXT A
17Ø GOTO 4Ø
```

Man überlege, wie das Ausdrucken von ähnlichen Dreiecken, wie 3–4–5, 6–8–1Ø, 9–12–15 etc., vermieden werden kann. Hier die Ausgabe eines entsprechend abgeänderten Programms.

```
RA#C1       1Ø:ØØ    WIEN Ø5/11/71

RECHTWINKELIGE DREIECKE
MIT GANZZAHLIGEN SEITENLAENGEN
```

```
KATHETENLAENGE
MIN: 3          MAX: 25

 A              B              C
 3              4              5
 5              12             13
 7              24             25
 8              15             17
 20             21             29

OUT OF DATA  IN 40

USED     3.33 UNITS.
```

Aufgabe 2 (Angaben S. 57)

Es müssen PRINTUSING und zwei Masken verwendet werden.

```
RA#01      12:07    WIEN  05/10/71

1:##.##
2:###
10 READ A,B
20 PRINTUSING 1,A;
30 PRINTUSING 2,B
40 GOTO 10
50 DATA -1.23,-45,6.78,90,35.711,-1317
```

Aufgabe 3 (Angaben S. 57)

30 ON SGN (V) +2 GOTO 100, 200, 300

Anwendung in einem Programm:

```
RA#003     12:20    WIEN  24/09/71

10 READ V
20 PRINT V,
30 ON SGN(V)+2 GOTO 100,200,300
100 PRINT "V < 0"
110 GOTO 10
200 PRINT "V = 0"
210 GOTO 10
300 PRINT "V > 0"
310 GOTO 10
400 DATA -1.23,0,4.56
```

```
RA#003     12:21    WIEN  24/09/71

-1.23            V < 0
 0               V = 0
 4.56            V > 0

OUT OF DATA  IN 10



USED       1.50 UNITS.
```

Eine andere Anwendungsversion:

```
RA#004     14:07    WIEN  03/11/71

1 :##.####↑↑↑↑     V ' 0
2 V$(1)="<"
3 V$(2)="="
4 V$(3)=">"
10 READ V
20 PRINTUSING 1,V,V$(SGN(V)+2)
30 GOTO 10
40 DATA -1.23,0,1.23



RA#004     14:08    WIEN  03/11/71

-1.2300E+00      V < 0
 0.0000E+00      V = 0
 1.2300E+00      V > 0

OUT OF DATA  IN 10



USED       1.33 UNITS.
```

Aufgabe 4 (Angaben S. 57)

Zwei (von vielleicht mehreren) Möglichkeiten sind:

```
200 IF SGN (U-X) =SGN (0-X) THEN 500
220 ...
```

oder

```
200 IF ABS (X-(0+U)/2) > ABS ((0-U)/2) THEN 500
220 ...
```

Die zweite Lösung ist, in der gegebenen Form, ein schlechter Ersatz für die ursprünglichen zwei Anweisungen. Es sind 39+3 statt 2·(15+3) Zeichen notwendig, und der rechnerische Aufwand ist groß. Diese Lösung wird jedoch dann einfach, wenn 0 und U numerisch fest vorgegeben sind (Z.B. U=10 0=70):

```
200 IF ABS (X-40) > 30 THEN 500
220 ...
```

Man beachte: Für U=0=X führt die erste Lösung nach 500, die zweite nach 220.

Aufgabe 5 (Angaben S. 58)

Das Problem wird durch Zählen in modulo gelöst. n (mod m) ist definiert als

$$n - m \cdot \langle n/m \rangle$$

wobei $\langle x \rangle$ größte ganze Zahl kleiner x bedeutet.

Zählen von 0 bis 10 modulo 3, z.B., verglichen mit normalem Zählen:

normal	0	1	2	3	4	5	6	7	8	9	10
modulo 3	0	1	2	0	1	2	0	1	2	0	1

Ist also eine Zahl n ein ganzes Vielfaches einer Zahl m, dann ist n (mod m) = 0. (D.h. es bleibt bei Division kein Rest). Diese Tatsache ist in 80 und 100 programmiert.

```
RA#U        15:54    WIEN   06/10/71

10 N=14
20 M=3
30 K=2
40 FOR I=1 TO N
50 L=L+1
60 PRINT I
70 R=L/M
80 IF R<>INT(R) THEN 120
90 PRINT
100 IF R/K<>INT(R/K) THEN 120
110 PRINT
120 NEXT I
```

Aufgabe 6 (Angaben S. 59)

1Ø...Initialisierung der RND-Funktion
3Ø...Bildung einer Zufallszahl zwischen Ø und 1ØØ
4Ø...Ausdrucken der Zahl und Vorrücken zu Druckposition 7 (die Funktion TAB (Ausdruck) ist nicht in allen TS-BASIC-Systemen eingebaut)
5Ø... Prüfen, ob Zahl gerade oder ungerade ist. Sprung zu 7Ø und damit Überspringen von 6Ø, wenn gerade.
6Ø...Drucken von UN und Stehenbleiben

```
RA#O        10:28    WIEN 02/06/71

10 Z=RND(-1)
20 FOR I=1 TO 10
30 Z=INT(101*RND(0))
40 PRINT Z;TAB(7);
50 IF Z/2=INT(Z/2) THEN 70
60 PRINT "UN";
70 PRINT "GERADE"
80 NEXT I
```

Aufgabe 7 (Angaben S. 59)

1Ø...da mehr als 1Ø $-Variable gebraucht werden, muß T$ dimensioniert werden
2Ø...5Ø...die 15 Buchstaben werden eingelesen
6Ø...8Ø...Ausdrucken des Textes. Man beachte Leerstelle (Zwischenraum) in "TABELLE "

```
RA#M         9:12    WIEN 07/06/71

10DIMT$(15)
20FORJ=1TO15
30READT$(J)
40NEXTJ
50DATAA,B,C,D,E,F,G,H,I,J,K,L,M,N,O
60FORI=1TO15
70PRINT"TABELLE ";T$(I)
80NEXTI
```

Aufgabe 8 (Angaben S. 60)

10,20...da die letzten fünf Buchstaben der beiden Worte gleich sind, wurde zur Platzeinsparung nur der erste Teil als alphamerische Variable definiert.

Man beachte die Verschiebung um eine Stelle in der letzten Zeile der Ausgabe. N ist in diesem Fall gleich 100, die Maske ist aber nur für zweiziffrige N geplant.

```
RA#W        10:18    WIEN 31/08/71

1:ORDNEN VON ### ZAHLEN IN "    "ENDER FOLGE
10 W$(1)="STEIG"
20 W$(2)=" FALL"
30 READ N,J
40 PRINTUSING 1, N,W$(J)
50 PRINT
60 GOTO 30
70 DATA 9,1,9,2,20,1,100,2
```

Aufgabe 9 (Angaben S. 60)

1:...die Variablennamen werden zweistellig definiert. Die einstelligen Namen werden durch Vorsetzen eines Blank zweistellig gemacht.

40...PA und PB müssen im Programm neue Namen bekommen.

```
RA#X        10:15    WIEN 31/08/71

1:""=##.#######   ""=##.#######   ""=##.#######
10 READ X,Y,Z,A1,A2,A3,L,P1,P2
20 PRINTUSING 1," X",X," Y",Y," Z",Z
30 PRINTUSING 1,"A1",A1,"A2",A2,"A3",A3
40 PRINTUSING 1," L",L,"PA",P1,"PB",P2
50 PRINT
60 GOTO 10
70 DATA 1.234,5.678E9,.0123E-4,5,-6,789,0,12E34,5E-67
```

Aufgabe 10 (Angaben S. 61)

10, 20...liest die Drucksymbole ein. Man beachte, wie eine Leerstelle eingelesen werden muß.

30...J steuert die Anzahl der "Felder" vertikal

40...R steuert die Anzahl der Reihen die ein "Feld" ausmachen

50...I steuert die Anzahl der "Felder" horizontal

70...C steuert die Anzahl der Spalten die ein "Feld" ausmachen

60...N ist I+J modulo 2, d.h. N=0, wenn I+J gerade, N=1, wenn I+J ungerade. Dadurch wird das "Schachbrettmuster" gesteuert.

```
RA#P        13:25    WIEN 02/06/71

1 REM SCHACHBRETT
10 READ C$(0),C$(1)
20 DATA *," "
30 FOR J=0 TO 3
40 FOR R=0 TO 2
50 FOR I=0 TO 3
60 N=(I+J)-2*INT((I+J)/2)
70 FOR C=0 TO 4
80 PRINT C$(N);
90 NEXT C
100 NEXT I
110 PRINT
120 NEXT R
130 NEXT J
```

Aufgabe 11 (Angaben S. 62)

Die DEF-Anweisung lautet

10 DEF FNP (T) = A* (SGN (SGN (T−T1) +1)−SGN (SGN (T−T2) +1)) *SIN (0*T)

Hier ist T1 der Zeitpunkt des Beginns des Pulses, T2 das Ende (also T2 – T1 die Dauer des Pulses), 0 steht für $2\pi f$.

Soll nur ein Einschaltvorgang (ohne darauffolgendes Ausschalten) betrachtet werden, also $u(t-t_0)$, dann wird nur ein Teil des Ausdruckes genommen. Zum Beispiel

10 DEF FNS (T) =SGN (SGN (T−T0) +1) *SIN (0*T)

Die erstangegebene DEF-Anweisung wurde im Programm RA#S verwendet. Zusätzliche Informationen im Programm RA#S:

```
10 DEF FNY(T)=A0*(SGN(SGN(T-T1)+1)-SGN(SGN(T-T2)+1))*SIN(0*(T-T1))
12 T1=3
14 T2=35
16 0=3.14159265/8
18 A0=2
30 DATA 0,40,1
50 DATA 7.5,35,*
```

Aufgabe 12 (Angaben S. 64)

S$(I) ist das römische Zahlensymbol, W(I) die dazugehörige arabische Zahl. Algorithmus: Es werden nicht nur die "Grundzahlen", sondern auch sämtliche durch "Vorstellen" entstehenden Zahlen eingelesen. Damit kann man die Zahl einfach von links nach rechts aufbauen.

100...sichert gegen die Eingabe von Dezimalstellen. Programm ändert Eingabe und druckt geänderten Wert aus (ohne Meldung der Änderung!).
120...prüft, ob Zahl innerhalb des vorgegebenen Bereichs liegt
150...200...beginnend bei der größten Grundzahl (1000), wird versucht, ob diese von der gegebenen Zahl abgezogen werden kann. Wenn nicht, wird nächstkleinere "Grundzahl" probiert. Wenn ja, wird das zugehörige Symbol ausgedruckt (170), der dem Symbol entsprechende wirkliche Wert abgezogen und erneut geprüft, ob die gleiche Grundzahl nochmals abgezogen werden kann (190, 160). Schließlich wird die Zahl Null, und die Schleife nach dem zwölften Durchlauf verlassen.
210...nach Verlassen der Schleife muß in neue Zeile geschaltet werden, denn die Schleife läßt den Schreibkopf irgendwo in der Zeile stehen.
220...zur nächsten umzuwandelnden Zahl.

```
RA#I       10:33    WIEN 02/06/71

10 DIM W(12),S$(12)
20 FOR I=0 TO 12
30 READ S$(I),W(I)
40 NEXT I
50 DATA M,1000,CM,900,D,500,CD,400,C,100,XC,90
60 DATA L,50,XL,40,X,10,IX,9,V,5,IV,4,I,1
70 PRINT "UMWANDLUNG ARAB. IN ROEM. ZAHLEN"
80 PRINT
90 READ Z
100 Z=INT(Z)
110 PRINT Z,
120 IF ABS(Z-2001)<=2000 THEN 150
130 PRINT "NUR 1...4001"
140 GOTO 90
150 FOR I=0 TO 12
160 IF Z<W(I) THEN 200
170 PRINT S$(I);
180 Z=Z-W(I)
190 GOTO 160
200 NEXT I
210 PRINT
220 GOTO 90
230 DATA 3888,276,2555,444,999,777,331,95,-5,4002,4001,1971
```

Aufgabe 13 (Angaben S. 64)

Die Ausgabe wurde so definiert, daß alle 72 Druckpositionen verwendet werden müssen.

2:...in der letzten Stelle sind drei # übereinandergedruckt, da in der Maske Zeilennummer und : zwei Stellen wegnehmen.

In der X=1 Zeile, Spalte für 1/X, sind scheinbar nur vier Dezimalstellen vorhanden. Tatsächlich werden fünf gedruckt, aber die letzten beiden übereinander. Dies, weil in der Maske keine Einerstelle vorgesehen ist, für X=1 aber eine vorhanden ist und daher nach rechts erweitert wird. Eine 73ste Druckposition gibt es aber nicht, daher das Übereinanderdrucken.

```
RA#Z        10:24    WIEN 31/08/71

1:  X    X↑2    X↑3    X↑1/2     X↑1/3     LN(X)     LG(X)       EXP(X)       1X
2:###   #### ##### ##.##### ##.##### ##.##### ##.##### ##.#####↑↑↑↑ #.###
10  PRINTUSING 1
20  FOR X=1 TO 10
30  X2=X*X
40  X3=X2*X
50  W2=SQR(X)
60  W3=X↑(1/3)
70  L1=LOG(X)
80  L2=.434294482*L1
90  E=EXP(X)
100 R=1/X
110 PRINTUSING 2, X,X2,X3,W2,W3,L1,L2,E,R
120 NEXT X
```

```
RA#Z        10:23    WIEN 31/08/71

  X    X↑2    X↑3    X↑1/2     X↑1/3     LN(X)     LG(X)       EXP(X)       1/X
  1      1      1  1.00000   1.00000    .00000    .00000  2.71828E+00  1.0000
  2      4      8  1.41421   1.25992    .69315    .30103  7.38906E+00   .50000
  3      9     27  1.73205   1.44225   1.09861    .47712  2.00855E+01   .33333
  4     16     64  2.00000   1.58740   1.38629    .60206  5.45982E+01   .25000
  5     25    125  2.23607   1.70998   1.60944    .69897  1.48413E+02   .20000
  6     36    216  2.44949   1.81712   1.79176    .77815  4.03429E+02   .16667
  7     49    343  2.64575   1.91293   1.94591    .84510  1.09663E+03   .14286
  8     64    512  2.82843   2.00000   2.07944    .90309  2.98096E+03   .12500
  9     81    729  3.00000   2.08008   2.19722    .95424  8.10308E+03   .11111
 10    100   1000  3.16228   2.15443   2.30259   1.00000  2.20265E+04   .10000



USED      2.83 UNITS.
```

Aufgabe 14 (Angaben S. 65)

10...30...Titel
40...liest ein Zahlenpaar ein
60...Ausgabe der Daten
70...100...Kern des Programms. Kann für wiederholte Verwendung in einem Hauptprogramm als Unterprogramm (GOSUB) eingesetzt werden.
70...bestimmt den Betrag (R)
80...macht den Winkel (P) + oder – $\pi/2$, abhängig vom Imaginärteil
90...wenn der Realteil 0 ist, ist P von 80 das Ergebnis
100...es wird ATN berechnet und im zweiten Teil der Anweisung durch Hinzufügen von π in den richtigen Quadranten verlegt.
110...Ausgabe des Ergebnisses mit Umrechnung des Winkels von Bogenmaß in Grad-maß
130...zu Einlesung neuer Daten

```
RA#F        12:47    WIEN 26/05/71

10 PRINT "TRANSFORMATION VON"
20 PRINT "KART. KOORD. IN POLAR KOORD."
30 PRINT
40 READ X,Y
50 DATA 0,0,0,2,0,-3,4,0,-5,0,-.75470958,.65605903
55 DATA -.32556815,-.94551858
60 PRINT "X=";X,"Y=";Y
70 R=SQR(X↑2+Y↑2)
80 P=SGN(Y)*1.57079633
90 IF X=0 THEN 110
100 P=ATN(Y/X)+(1-SGN(X))*SGN(SGN(Y)+.5)*1.57079633
110 PRINT "R=";R,"P=";P*57.2957795
120 PRINT
130 GOTO 40
```

Aufgabe 15 (Angaben S. 66)

Algorithmus: Der Winkel *B* wird in Sekunden umgerechnet (gerundet auf Zehntel-Sekunden). Durch Division mit 60 werden Minuten und Grade gefunden.

Man beachte in der vor- und drittletzten Zeile das unterschiedliche Ergebnis für scheinbar gleiche Eingabe. Tatsächlich unterscheiden sich die beiden Eingaben um 5 in der neunten Stelle, wie 100 DATA (letzte Eintragung) und 110 DATA (erste Eintragung) zeigt. Die neunte Stelle wird zwar intern verwendet, kann aber nicht ausgegeben werden.

```
RA#LA        8:02    WIEN 05/05/71

1:###.##########[RAD] = ####[O]###" "###.#"  "
2 READ M$,S$
3 DATA ['],["]
10 READ B
20 S=INT(2062648.06*B+.5)/10
30 M=INT(S/60)
40 S=S-60*M
50 G=INT(M/60)
60 M=M-60*G
70 PRINTUSING 1,B,G,M,M$,S,S$
80 GOTO 10
90 DATA 1.57079633,3.14159265,6.28318531,1.74532925E-2
100 DATA .290888209E-3,.484813681E-5,1,174.53292
110 DATA 174.532925,1745.32925
```

Aufgabe 16 (Angaben S. 66)

20...Initialisierung des Feldes A
60...Vorrücken zum Beginn des "Dreiecks"
70...90...Aufbauen der Zeile von rechts nach links. Es werden N−1 Elemente aus den N−1 Elementen der darüberliegenden Zeile errechnet. Das N−te Element ist gleich dem Ø−ten Element (1ØØ)
110...130...Herausdrucken der N Elemente

Man beachte: A(I) hat die Dimension 1Ø. Bei DIM(1Ø) (nicht notwendige Anweisung, da automatisch für 1Ø dimensioniert wird) können 11 Elemente (einschließlich A(Ø)) gespeichert werden. Daher ist bis N=1Ø (11 Elemente) keine DIM-Anweisung notwendig.

```
RA#J         12:58    WIEN 26/05/71

1:###
2:#######
10 PRINT "PASCAL' DREIECK"
20 A(Ø)=1
30 PRINT
40 FOR N=Ø TO 10
50 PRINTUSING 1,N,
60 PRINT TAB(33-3*N);
70 FOR I=N-1 TO 1 STEP -1
80 A(I)=A(I)+A(I-1)
90 NEXT I
100 A(N)=A(Ø)
110 FOR I=Ø TO N
120 PRINTUSING 2,A(I),
130 NEXT I
140 PRINT
150 NEXT N
```

Aufgabe 17 (Angaben S. 67)

Besprechung:

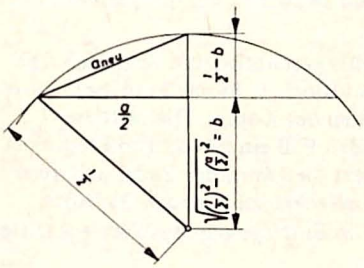
10, 20...Definition von $-Konstanten für Überschriften
50...Einlesen von N
60...N wird positiv und ganzzahlig erzwungen und um 1 erniedrigt (Erniedrigung um 1 zur Vereinfachung der Adressierung von A(0))
70...nur Werte zwischen 2 und 11, einschließlich, sind zugelassen
80, 90...Fehlermeldung und Stop
100...120...Einlesen der ersten N Glieder
130...$-Variable wird auf " ERS" gesetzt (siehe 10)
140...im Unterprogramm 500 bis 560 werden Kopf und N Glieder ausgedruckt
160...260...maximal 100 mal zu durchlaufende Schleife
170...220...Bildung eines neuen Gliedes. B ist temporäre Speicherstelle, denn bei Bildung der Summe wird gleichzeitig im Feld A nach links verschoben (200)
230...ein neues Verhältnis wird gebildet
240...wenn Verhältnis genau genug, Sprung aus der Schleife
250...das neue Verhältnis wird zum alten gemacht
270...290...Meldung, daß Genauigkeit nicht erreicht
300...ähnlich 130, aber nun Zuteilung des Wertes "LETZ"
310...wie 140
320...Ausdrucken von Text und Wert des Verhältnisses
340...Rücksprung zum Einlesen neuer Daten

```
RA#V       14:40    WIEN  24/09/71

10 A$="  ERS"
20 E$=" LETZ"
30 PRINT "VERALLGEMEINERTE FIBONACCI-FOLGE"
40 PRINT
50 READ N
60 N=INT(ABS(N))-1
70 IF ABS(N-5.5)<=4.5 THEN 100
80 PRINT "N=";N+1;"  2<=N<=11"
90 STOP
100 FOR I=0 TO N
110 READ A(I)
120 NEXT I
130 Z$=A$
140 GOSUB 500
160 FOR K=1 TO 100
170 B=A(0)
180 FOR I=1 TO N
190 B=B+A(I)
```

```
200 A(I-1)=A(I)
210 NEXT I
220 A(N)=B
230 R1=A(N)/A(N-1)
240 IF ABS(R1-R)<=1E-6 THEN 300
250 R=R1
260 NEXT K
270 PRINT "A(N)/A(N-1) NACH 100 GLIEDERN"
280 PRINT "NOCH NICHT KONSTANT"
290 PRINT
300 Z$=E$
310 GOSUB 500
320 PRINT "A(N)/A(N-1)=";R1
330 PRINT
340 GOTO 40
500 PRINT "DIE";Z$;"TEN";N+1;"GLIEDER"
510 FOR I=0 TO N
520 PRINT A(I),
530 NEXT I
540 PRINT
550 PRINT
560 RETURN
1000 DATA 2,0,1,5,-3,10,54,-33,14
1010 DATA 11,0,1,2,3,4,5,6,7,8,9,10
```

Aufgabe 18 (Angaben S. 68)

Die jeweils neue Seitenlänge folgt aus der alten mit Hilfe des Pythagoräischen Lehrsatzes. (Siehe unten)



$$a_{neu} = \sqrt{(\tfrac{a}{2})^2 + (\tfrac{1}{2} - b)^2}$$

$$a_{neu} = \tfrac{1}{2}\sqrt{a^2 + (1 - 2b)^2}$$

$$2b = \sqrt{1 - a^2} \qquad a = a_{alt}$$

Besprechung:

13Ø, 14Ø...initialisiert mit Quadrat
16Ø...verdoppelt Seitenzahl
17Ø...a^2 zur Verwendung in 18Ø und 19Ø wird berechnet
18Ø...Errechnung von 2b (im Programm X genannt)
19Ø...Errechnung der neuen Seitenlänge
2ØØ...Ausdrucken der gewünschten Information

Ab 16384 bleibt der Wert für π unverändert, weil $a^2 \ll 1$ und der Unterschied zwischen 1 und $1-a^2$ im Computer nicht mehr dargestellt werden kann. Das führt zu $a_{neu} = a_{alt}/2$ in 19Ø und damit zu unverändertem Wert für π, da andererseits $F_{neu} = 2 \cdot F_{alt}$.

```
RA#B1        9:11    WIEN 19/Ø5/71

1ØØ:#######     ##.#######
11Ø PRINT "PI UEBER 2N-ECK"
12Ø PRINT
13Ø F=4
14Ø A=1/SQR(2)
15Ø FOR I=1 TO 14
16Ø F=2*F
17Ø A2=A*A
18Ø X=SQR(1-A2)
19Ø A=.5*SQR(A2+(1-X)↑2)
2ØØ PRINTUSING 1ØØ,F,F*A
21Ø NEXT I
```

Aufgabe 19 (Angaben S. 69)

I ist der Index vertikal, *J* der Index horizontal. Die y-Werte sind in den großen *D(I,J)*-Block aufgenommen als *D(I, Ø)*.

Die *D(I,J)*-Berechnung (8Ø...12Ø) erfolgt spaltenweise (der Index *I* der inneren Schleife läuft rasch); immer beginnend bei der Differenz *D(L,L)*. Dadurch wird der dreieckigen Form der Differenzentafel Rechnung getragen. (DIM D(2Ø,2Ø), ein quadratisches Feld reservierend, ist daher sehr verschwenderisch).

Das Ausdrucken der Tabelle erfolgt in drei ineinander geschachtelten Schleifen. Die äußerste (K-) Schleife steuert die in der Angabe mit Block 1, Block 2 etc. bezeichneten Teile der Tabelle, in ihr liegt also das Ausdrucken des Kopfes. Die mittlere (I-) Schleife steuert den Zeilenindex. In ihr wird auf jeden Fall ein y-Wert (im Programm D(I,Ø)) ausgedruckt. Die innerste (J-) Schleife steuert die Länge der Zeile, und zwar über zwei Kriterien : 1) Durch J=... TO I wird die Dreiecksform erzeugt. 2) Durch IF J > = K+7 wird die Zeile auf jeden Fall nach neun Eintragungen (y-Wert + 8 Differenzen) abgebrochen und eine neue Zeile begonnen.

Man beachte: Es ist in einem richtigen Programm nicht sehr sinnvoll, Differenzen so hoher Ordnung wie im Übungsbeispiel zu berechnen. Wie aus der Tabelle ersichtlich, sind die Differenzen höherer Ordnung (im Beispiel ab etwa 1Ø) vollkommen unbrauchbar.

Bemerkung: Die Blankstelle zwischen den Spalten ist nicht direkt programmiert. Sie wird durch Ausnützen der Eigenschaft von PRINTUSING, Nullen links des Dezimalpunktes bei Zahlen < 1 zu unterdrücken, erreicht. (Ein mögliches negatives Vorzeichen wird dann unmittelbar vor dem Dezimalpunkt gedruckt). Da alle Differenzen < 1 sind, wird so "kostenlos" eine Blankstelle erzeugt.

```
RA#H       12:45    WIEN 05/06/71

1:##.#####
2:      Y      D###      D###      D###      D###      D###      D###      D###      D##
10 DIM D(20,20)
20 PRINT "DIFFERENZEN-TAFEL"
30 PRINT
40 N=20
50 FOR I=0 TO N
60 D(I,0)=SIN(.31415926#I)
70 NEXT I
80 FOR J=1 TO N
90 FOR I=J TO N
100 D(I,J)=D(I,J-1)-D(I-1,J-1)
110 NEXT I
120 NEXT J
130 FOR K=1 TO N STEP 8
140 PRINTUSING 2,K,K+1,K+2,K+3,K+4,K+5,K+6,K+7
150 FOR I=K-1 TO N
160 PRINTUSING 1,D(I,0),
170 FOR J=K TO I
180 PRINTUSING 1,D(I,J),
190 IF J>=K+7 THEN 210
200 NEXT J
210 PRINT
220 NEXT I
230 PRINT
240 NEXT K
```

7. Anhang zu Mark I – BASIC

7.1 Überblick über Fehlermeldungen

Die vom Compiler bei der Diagnostik ausgegebenen Fehlermeldungen haben einen unschätzbaren Wert bei Erstellung, Testen und Anwendung eines Programms. Im folgenden sollen die häufiger auftretenden Fehlermeldungen des General Electric Betriebs-Systems Mark I/7 kurz besprochen werden. Die Liste erhebt keinen Anspruch auf Vollständigkeit.

Grundsätzlich kann man zwei Gruppen von Fehlermeldungen unterscheiden: Solche die sich auf Systembefehle, und solche, die sich auf Programmbefehle beziehen. Bei den Programmbefehlen gibt es zwei weitere Untergruppen, und zwar Meldungen, die gleich bei der Compilierung (d.h. Übersetzung) ausgegeben werden, und Meldungen, die erst während des Laufes ausgelöst werden. (Zn...Zeilennummer)

Beispiele für Fehlermeldungen bei Systembefehlen:

FILE CURRENTLY SAVED–Name

Wenn SAV verwendet wurde, obwohl ein Programm dieses Namens im Bibliotheksspeicher vorhanden ist.

XXX NOT AVAILABLE

Wenn eine Programmiersprache, die im System nicht vorgesehen ist, verlangt wird.

WHAT?

Reaktion auf Rechtschreibfehler bei Eingabe von Systembefehlen.

Beispiele für Compilier-Fehlermeldungen:

FOR WITHOUT NEXT

Bei, z.B., FOR A=..... NEXT K, oder wenn NEXT fehlt. Auch bei Überschneidungen von FOR-Schleifen.

ILLEGAL FORMULA IN Zn
ILLEGAL INSTRUCTION IN Zn
INCORRECT FORMAT IN Zn

Bei allgemeinen Formfehlern. Z.B.: A+/B oder FER A=... oder X(I,J,K) oder V$(3,4)

ILLEGAL RELATION IN Zn

Bei Formfehlern in der Bedingung von IF-THEN-Anweisungen.

NEXT WITHOUT FOR IN Zn

Wenn die FOR-Anweisung fehlt oder einen Formfehler hat.

NO DATA

Wenn das Programm eine READ-Anweisung, aber keine DATA-Anweisung enthält.

UNDEFINED IMAGE Zn

Wenn in einer PRINTUSING–Anweisung eine Zn verwendet wird die es nicht gibt oder keinen : hat.

UNDEFINED NUMBER Zn

Wenn es die Zeilennummer in IF-THEN Zn, GOTO Zn oder ON – GOTO Zn1, Zn2.. nicht gibt oder, wenn sie zu einem nicht ausführbaren Befehl (z.B. DIM) gehört.

Beispiele für Lauf-Fehlermeldungen:

BAD IMAGE IN Zn

Bei Fehlern in Zusammenhang mit Zeilenmasken. Z.B., in der Maske folgt ein " "-Feld (alphamerisches Feld), in der PRINTUSING-Liste aber eine numerische Variable. Führt zu Programmstop.

DIVISION BY ZERO IN Zn

Z.B. bei A=B/C, wenn C nicht definiert ist. Es wird mit der größten darstellbaren Zahl weitergerechnet.

OUT OF DATA IN Zn

Die Daten in der DATA-Anweisung wurden schon alle gelesen. Zn ist die Zn der READ-Anweisung die keine Daten mehr vorfand. Führt zu Programmstop.

OVERFLOW IN Zn

Eine Rechenoperation führte zu einer Zahl, die absolut größer ist als die größte darstellbare Zahl. Es wird mit der größten darstellbaren Zahl weitergerechnet.

RETURN BEFORE GOSUB OR CALL IN Zn

Wenn Anweisungen die logisch zu RETURN führen, nicht durch GOSUB (z.B.), sondern GOTO (z.B.) oder geraden Programmfluß (z.B.) erreicht werden. Führt zu Programmstop.

SUBSCRIPT ERROR IN Zn

Wenn ein Wert des Index errechnet wird, der > 10 ist (ohne DIM-Anweisung) oder größer als in DIM gegeben, bzw. wenn der Wert des Index negativ ist. Führt zu Programmstop.

UNDERFLOW IN Zn

Eine Rechenoperation führte zu einer Zahl, die absolut kleiner ist als die kleinste darstellbare Zahl. Es wird mit Null weitergerechnet.

7.2 BCD-Code (Oktal)

0	00	A	21	J	41	/	61
1	01	B	22	K	42	S	62
2	02	C	23	L	43	T	63
3	03	D	24	M	44	U	64
4	04	E	25	N	45	V	65
5	05	F	26	O	46	W	66
6	06	G	27	P	47	X	67
7	07	H	30	Q	50	Y	70
8	10	I	31	R	51	Z	71
9	11	.	33	$	53	,	73
'	12	"	34	*	54	)	74
:	13	?	35	>	56	[	75
(	14	<	36	↑	57	]	76
)	15	-	40		60		
=	16						
#	17						
+	20						

7.3 Numerische Konstanten

π	3.14159265	$1/\pi$	Ø.318Ø9886
2π	6.28318531	$1/2\pi$	Ø.159154943
$\pi/2$	1.57079633	$2/\pi$	Ø.636619772
4π	12.5663706	$1/4\pi$	Ø.Ø795774715
$\sqrt{2}$	1.41421356	$1/\sqrt{2}$	Ø.7Ø71Ø6781
$\sqrt{3}$	1.732Ø5Ø81	$\sqrt{1Ø}$	3.16227766
$\sqrt[3]{1Ø}$	2.15443469	$\sqrt[3]{1ØØ}$	4.64158883
e	2.71828183	lg e	Ø.434294482
ln 1Ø	2.3Ø258509	ln 2	Ø.693147181
1 rd	57.2957795^0	1^0	$1.74532925.1Ø^{-2}$rd
1 rd	3437.74677'	1'	$2.9Ø888Ø9.1Ø^{-4}$rd
1 rd	2Ø6264.8Ø6"	1"	$4.84813681.1Ø^{-6}$rd

Im Betriebs-System Mark I/7 von General Electric:

Größte darstellbare Zahl: ± 5.7896Ø45E+76
Kleinste darstellbare Zahl ≠ Ø: ± 4.318Ø85E–78
Größte ganze Zahl: ± 1Ø73741823 = ± $(2^{3Ø} - 1)$.

8. Mark III-BASIC

In diesem Kapitel gilt: Ø für den Buchstaben o
0 für die Ziffer Null
ƀ für Blank (Leerstelle), wenn notwendig.

8.1 Allgemeines

Mark III bietet im Dialogbetrieb wesentlich mehr Möglichkeiten; so z.B. eine weitgehende Selbstverwaltung des TS-Betriebes. Durch den Einsatz größerer Computer bei Mark III gibt es kaum eine Beschränkung der Programmgröße und die Ausführung ist zehn- bis zwanzigmal schneller als in Mark I. Die Schnelligkeit, vereint mit der neuen Organisation des Betriebssystems, macht ein Warten in der „Schlange" kaum feststellbar. Sonstige Programmierhilfen, z.B. Compilier- und Lauf-Fehlermeldungen, sind so ausführlich wie in Mark I und zum Teil sogar erweitert. Hand in Hand mit dem Ausbau des Betriebssystems ging auch der Ausbau der Programmiersprachen, und zwar sowohl von BASIC als auch von FØRTRAN IV.

Mark III-BASIC ist gegenüber Mark I-BASIC wesentlich erweitert worden. So stehen z.B. in Mark I nicht vorhandene Manipulationsmöglichkeiten mit alphamerischen Daten zur Verfügung; eine besonders für den kommerziellen Bereich interessante Erweiterung des Anweisungsrepertoirs. Auf dem technischen Sektor ist es besonders das mehrzeilige Funktionsunterprogramm welches eine gewisse Schwäche des Mark I-BASIC eliminiert.

Nun ist es aber im Rahmen eines Kapitels nicht möglich alles neu Angebotene zu besprechen. Da jedoch nicht alle Anweisungen mit der gleichen Häufigkeit verwendet werden, ist eine Besprechung der meistangewendeten Anweisungen ausreichend für einen sehr großen Prozentsatz von Gebrauchsprogrammen. Von den sich auf alphamerische Größen beziehenden Anweisungen wurden einige zu Demonstrationszwecken aufgenommen.

8.2 Zeichen

Ziffern, Buchstaben und Sonderzeichen wie in Mark I; bei den Sonderzeichen sind zusätzlich ein- und ausgebbar ! % & \ @.

8.3 Code

Der verwendete Code ist nicht mehr BCD, sondern ASCII [1]. Dieser Unterschied ist wesentlich bei der Verarbeitung alphamerischer Größen, besonders bei Vergleich mit IF. Ein Auszug dieses Codes befindet sich im Anhang.

8.4 Konstanten

Zahlenbereich (vereinfachte Grenzwerte): ± 1.5E–39 < = Z < = ± 1.7E+38
Genauigkeit: 8 Stellen
Größte Ganzzahlkonstante: ± 134 217 727 (= $2^{27}-1$)
Alphamerische Konstanten: Die Beschränkung auf 15 Zeichen entfällt (wie z.B. schon immer in PRINT bei Mark I). Maximale Länge 158 Zeichen, aber praktische Begrenzung meist durch Länge einer Zeile am Terminal gegeben.

8.5 Variablen

Zulässige Namen für numerische Variablen wie in Mark I.
Zulässige Namen für alphamerische Variablen: Buchstabe-Ziffer-$ zulässig. Z.B.: V3$
Zweifache Indizierung zulässig. Z.B.: A$ (M,N) (Aber nicht AI$ (M,N))

8.6 Eingebaute Funktionen (zusätzliche, bzw. unterschiedliche, zu den in Mark I zulässigen Funktionen)

CØT(X) cotan x (x in Radians)

RAN startet Zufallszahlengenerator mit Zufallszahl (ähnlich RND (–1) in Mark I). Z.B.: 10 RAN

RND erzeugt eine neue Zufallszahl (RND(0) in Mark I). 0 und 1 werden nicht erzeugt. Z.B.: 50 Z=10* RND

REP$ (A$, B$, C$, I, J) ersetzt in der alphamerischen Größe A$, den Teil B$ durch die Größe C$ und zwar für I-maliges Auffinden von B$, mit Suchbeginn an der J-ten Stelle von A$.

```
Z.B.:   10 A$="A+B+C+D"
        20 B$="+"
        30 D$=REP$(A$, B$, "–", 1,4)
        40 PRINT D$
gibt:   A+B–C+D
```

3 anstelle von 4 in 30 hätte das gleiche Ergebnis geliefert.
2 anstelle von 1 in 30 hätte auch das + zwischen C und D zu – geändert.
I=–1 ersetzt *alle* B$ mit C$, beginnend bei der J-ten Stelle.

SCN (A$, B$, I, J) sucht die Größe B$ in der Größe A$ und zwar für I-maliges Auf-

---

[1] American Standard Code for Information Interchange.

treten, beginnend bei der J-ten-Stelle. Als Ergebnis wird die Stelle des I-ten Auftretens von B$ nach der J-ten Stelle geliefert.

Z.B.: 10 A$="A+B+C+D"
20 P=SCN(A$, "+",1,3)
30 PRINT P

gibt: 4

4 anstelle von 3 in 20 hätte das gleiche Ergebnis geliefert.
2 anstelle von 1 in 20 hätte des Ergebnis 6 geliefert.
Besteht B$ aus mehreren Zeichen, liefert SCN die Stelle des ersten Zeichens von B$.

8.7 *Anweisungen* [1)]

END — *muß* die Anweisung mit der größten Zeilennummer sein. Z.B.: 999 END

' — Apostroph; für Kommentare

Z.B.: 100 S=S+R 'SUMME
150'SØRTIEREN

LET — ist auch in impliziter Form, d.h. mehrere Zuordnungen innerhalb einer Anweisung, möglich.

Z.B.: 10 PRINT W=2–SQR(X);SIN(A=6.28318531*W)
20 IF(N=K–5*INT(K/5))> Ø THEN 800
30 F=S+(H=H/2)
40 A(L,N=K–L)=(M=4+L)+A(I=L–1,N+1)
50 FØR I=(J=K+3) TØ (L=4–M) STEP (N=Ø)
60 ØN G=8*H GØTØ 200, 300, 400
70 DEF FNY(X)=CØS(Y=X/4)

Man beachte in 10 und 40: Weiter links erfolgte Zuordnungen können weiter rechts in der gleichen Zeile verwertet werden.

U.U. ist die implizite Zuordnung zwischen Klammern zu setzen und zwar immer dann, wenn in der Anweisung grundsätzlich ein =–Zeichen vorkommt oder vorkommen kann. (Siehe 20, 30 und 50.)
Nicht zulässig ist die implizite Form an, z.B., folgenden Stellen:

10 READ A=B
20 DIM A(I=5)
30 FØR (I=J) = 0 TØ 5
40 DEF FNY(X=T) = A+EXP(X)

A$+B$ — einfache Aneinanderreihung von alphamerischen Größen ist zulässig.

Z.B.: 10 A$="BASIC"
20 B$="PRAXIS"
30 PRINT A$+"–"+B$

gibt: BASIC–PRAXIS

PRINT — bei Verwendung von Strichpunkt als Trennzeichen wird hinter einer numerischen Ausgabe nur *ein* Blank erzeugt, unabhängig vom Wert

[1)] Siehe unter 9. für Anwendungen.

der Zahl. Blank anstelle von + und Ausgabe von maximal 6 Stellen wie in Mark I. Eine führende Null, wie z.B. in 0.123, wird nicht unterdrückt.
Z.B.: 100 PRINT 1;23;456;.789;–.987654321
gibt: ␣1␣␣23␣␣456␣␣0.789␣–0.987654␣
Kein Trennzeichen vor oder nach einer alphamerischen Konstante (eine Zeichenfolge zwischen Anführungszeichen) wird als Strichpunkt gesetzt interpretiert, wenn es sich nicht um das Ende der Parameterliste der PRINT-Anweisung handelt.
Z.B.: 120 PRINT Z "A" Y "B" TAB(Y) "*"
gibt für Z=5, Y=12:
␣5␣A␣12␣B␣␣␣*
Nach TAB ( ) ist ein Strichpunkt nach wie vor erforderlich, wenn nicht " folgt.
Z.B.: 10 PRINT TAB(N) K
20 PRINT TAB(L) A$
30 PRINT TAB (J) 5
alle drei Zeilen führen zu Fehlermeldungen
Die Steuerung für 15er Kolonnen[1] wurde verbessert. So druckt z.B.
100 PRINT ,"!", ,"+"
! in die Druckposition 15 und + in die Druckposition 45. Eine Folge von Strichpunkten (;;;) wird, wie zu erwarten, ignoriert.

MARGIN 72 verhindert ein Überdrucken in der 71sten Druckposition, bzw. ein Bedrucken der 72sten Druckposition, durch Weiterschalten in eine neue Zeile.
Z.B.: 5 MARGIN 30
verhindert ein Bedrucken der 30sten Druckposition.
Auszugebende Daten, numerischen oder alphamerischen Typs, werden ganz in die neue Zeile gedruckt (und nicht aufgespalten), wenn der MARGIN-Parameter (oben 30) größer als die Länge der auszugebenden Größe ist.

READ und DATA es werden getrennte Datenblöcke für numerische und alphamerische Daten erzeugt. Mischung von numerischen und alphamerischen Variablen in READ, und/oder numerischen und alphamerischen Konstanten in DATA beliebig, nur Sequenz innerhalb numerischer, bzw. alphamerischer Größen wesentlich.
Z.B.: 10 READ A, B, C$, D, E$, F$
20 DATA C, 1,2,E,F,3
liefert: A=1 B=2 D=3 C$=C E$=E F$=F
RESTØRE* setzt numerischen Datenblock zurück
RESTØRE$ setzt alphamerischen Datenblock zurück
RESTØRE setzt beide Datenblöcke zurück
} zum ersten Wert des Blocks

:–Anweisung Formate (Schablonen) für numerische Größen wie in Mark I (maximal 1+8 # – Zeichen), aber um neue Steuerungsmöglichkeiten erweitert. Unter anderem: Die Ausgabe von + (normal eine Leerstelle) kann be-

[1] bzw. 18er Kolonnen bei DØUBLE PRECISIØN (siehe später).

wirkt und gleichzeitig die Position des Vorzeichens fixiert werden, wenn man die numerischen Schablone mit einem oder mehreren + beginnt.

Z.B.: 1:+# # # ␣␣+ + + + . # #
2 PRINTUSING 1, 12.34, 12.34
gibt: +␣12␣␣+12.34

Formate für alphamerische Größen geändert gegenüber Mark I.

' markiert Beginn einer Schablone für alphamerische Größen (oder, allein stehend, Platz für *ein* alphamerisches Zeichen). Platz für die weiteren Stellen wird durch eine anschließende Folge gleicher Buchstaben reserviert. Unterbrechung der Folge beendet die Schablone. Folgende Buchstaben können verwendet werden,und haben die angegebene Bedeutung:

L linksbündig einsetzen; Schablone nicht erweitern.
Z.B.: 'LLLL wirkt wie "␣␣" in Mark I

R rechtsbündig einsetzen; Schablone nicht erweitern.
Z.B.: 'RRR HUN → ␣HUN HUNNE → HUNN

C centriere in gegebener Schablone; Schablone nicht erweitern.
Z.B.: 'CCC A → ␣A␣␣ AB → ␣AB␣

E linksbündig einsetzen; erweitern nach rechts, wenn die alphamerische Größe länger als die Schablone ist.
Z.B.: 'E U→ U␣ UV → UV UVW → UVW
Außer bei E wird in allen Fällen bei zu langen Größen rechts abgeschnitten.

PRINTUSING neben einer Zeilennummer ist nun auch eine alphamerische Variable oder alphamerische Konstante als Referenz für eine Maske zulässig.
Z.B.: 100 PRINTUSING "# # · # # # # # # ↑ ↑ ↑ ↑", R
200 A$="# # # ␣␣# # # #"
300 PRINTUSING A$, M, N

IF es sind alle Kombinationsmöglichkeiten zweier unterschiedlicher Bedingungssymbole zulässig.
Z.B.: 30 IF A = < B THEN 70
200 IF C > < 0 THEN 310
Zulässig ist auch GØTØ statt THEN, bzw. .EQ. statt =, .GE. statt >=, etc.
Unzulässig sind z.B.:
300 IF X ≪ THEN 30
400 IF Q < = > THEN 80

DEF ganz wesentlich erweitert, aber Mark I DEF-Regeln weiter gültig.
(DEF kann in Mark III auch auf alphamerische Größen angewendet werden; diese Möglichkeit wird jedoch im Weiteren nicht betrachtet)
Die Variablen in den unten angeführten Variablenlisten dürfen nur vom Typ numerisch und nicht indiziert sein.

Einzeiliges DEF:
DEF FNa (Variablenliste)=numerischer Ausdruck
Z.B.: 10 DEF FNA (B,C)=3*B+C↑2–Ø
20 DEF FND (E,F,G,H)=FNA(G,F)–FNA(E,H)

Mehrzeiliges DEF:
DEF FNa (Variablenliste) Variablenliste
Abschluß des mehrzeiligen DEF mit
FNEND
Z.B.: 100 DEF FNY(X,Y,K)I,T,V,M0 erste Zeile des Unterprog.
.
.
.
999 FNEND letzte Zeile des Unterprog.

Die Variablen in der Klammer sind als vom übergeordneten Programmteil (Hauptprogramm) festzulegende Funktionsparameter zu verstehen. Sie sind *Scheinvariablen* und können als solche bei Aufruf gleich oder auch anders als im Unterprogramm benannt werden; wesentlich ist nur ihre Reihenfolge.
Die Variablen der nicht geklammerten Liste sind *Lokalvariablen* und haben nur im entsprechenden Unterprogramm Gültigkeit. Die Verwendung gleicher Namen außerhalb des Unterprogramms ist ohne Wert- und Bedeutungs-Gleichheit möglich.
Variablen die in keiner der beiden Listen aufscheinen gelten als *Globalvariablen;* sie haben im Unterprogramm und im aufrufenden Programm gleichen Wert und gleichen Namen. Da in den Listen indizierte Variablen unzulässig sind folgt, daß indizierte Variablen globalen Charakter haben.
Der gewünschte Funktionswert wird im Unterprogramm durch
FNa=numerischer Ausdruck
erzeugt. Im aufrufenden Programmteil wird dieser Wert durch, z.B.
numerische Variable = FNa(Liste von numerischen Ausdrücken)
erhalten.
"Globalen" Charakter haben neben Feldern (indizierten Variablen), und daher DIM-Anweisungen, auch DATA- und DEF FN-Anweisungen, sowie :-Anweisungen.

Obwohl nicht ausdrücklich gefordert, ist es zur Vermeidung von sinnlosen Fehlermeldungen vorteilhaft, DEF-Unterprogramme an den Beginn des Gesamtprogramms zu setzen.

MAT-Anweisungen

MAT A=B zulässig (In Mark I-BASIC: MAT A=(1)*B)

MATPRINTUSING Zn, Liste Ausdrucken von Matrizen auch unter Formatkontrolle möglich. Zn ist die Zeilennummer einer Maske, entsprechend der jede Zeile der Matrix ausgedruckt wird. Die Liste enthält Namen von Matrizen.

Z.B.: 10 MATPRINTUSING 20,A
20: ## ·### ## ·### ## ·### ## ·### ## ·###

DET eine argumentlose Funktion, die den Wert der Determinante der zuletzt invertierten Matrix liefert.
Z.B.: 100 MAT I=INV(M)
110 PRINT "DETERMINANTE="; DET

DØUBLEPRECISIØN Als (erste)[1] Anweisung in einem Programm verwendet, bewirkt sie die Verarbeitung und Speicherung aller numerischer Größen im Programm als Doppelwort. (Bei einfacher Genauigkeit ist die Stellenzahl bei nicht-ganzen Zahlen 8,13; bei erhöhter Genauigkeit wird die Stellenzahl auf 18,96 erweitert.)
Bei Anwendung von PRINT werden für Ganzzahlen maximal 15 Stellen, für Fixkommazahlen maximal 12 signifikante Stellen und für Gleitkommazahlen maximal 10 signifikante Stellen ausgegeben. Die bei Verwendung von Beistrich als Trennzeichen gebildeten Kolonnen sind 18 (statt 15) Druckpositionen breit.
Bei Anwendung von PRINTUSING sind in der Zeilenmaske maximal 1+19 # -Zeichen pro numerischer Schablone zulässig.

\ Bei Verwendung von \ als Trennzeichen können in Mark III mehrere Anweisungen in eine Programmzeile [2] geschrieben werden. Die Gesamtzahl der Zeichen in einer Zeile darf aber 158 nicht Überschreiten.

Z.B.: 30FØRK=3TØ2↑LSTEP2\R=R+FNY(B+K*H)\NEXTK

Unzulässig sind weitere Zeilennummern links von einem Anweisungswort.
Z.B. 50 in der folgenden Programmzeile:
40PRINTM;N "*"\50PRINT\FØRI=1TØN
Daraus folgt, daß Anweisungen zu denen verzweigt werden kann eine neue Zeile einleiten müssen.
Anweisungsnummern in IF, GØTØ etc. sind jedoch zulässig. Auch die eigene Zeilennummer darf verwendet werden.
Z.B.: 60...\ØNKGØTØ200,60,120\...
Aus verschiedenen Gründen ist es günstig DATA-Anweisungen eigene Zeilen zuzuordnen.

Nach RAN ist eine Fortsetzung der Zeile mit \ nicht möglich (weitere Anweisungen in dieser Zeile werden ohne Meldung ignoriert).

---

[1] Nur ' (Apostroph) – und DIM-Anweisungen dürfen vor der DØUBLE PRECISIØN-Anweisung stehen.

[2] Eine mit einer Zeilennummer beginnende und mit RETURN/LINE-FEED endende Zeile (ein "record").

8.8 Systembefehle

RUN-1 Begrenzt Laufzeit auf eine CRU (Computer Resource Unit; praktisch eine Sekunde) RUN-3 würde auf drei CRU beschränken. Gewöhnliches RUN kann in Mark III bei logischen Kurzschlüssen (endlose Schleifen) teuer zu stehen kommen.

PURge Löscht mit REP oder SAV gespeicherte Programme. Werden mehrere Programmnamen mit einem PUR-Befehl gelöscht, sind die einzelnen Programmnamen durch Strichpunkt (nicht Beistrich) voneinander zu trennen.

"BREAK" Durch längeres Niederdrücken dieser Taste wird die Ausführung des letztgegebenen System-Befehls abgebrochen (nicht mehr durch CTRL/SHIFT/P, wie in Mark I).

CTRL+X Durch gleichzeitiges Drücken dieser beiden Tasten wird die Eingabe in dieser Zeile (ein System-Befehl oder eine Programmzeile) gelöscht. Das System antwortet mit DELETED.

CHA Zn/A$/B$/m* Beginnend mit der Zeile Zn wird die Zeichenfolge A$ durch die Zeichenfolge B$ ersetzt. Wie oben links geschrieben, werden die ersten m A$ ersetzt; wird nur * angeschlossen, werden alle A$ im Programm ersetzt; ohne m* wird nur das erstangetroffene A$ ersetzt.

Z.B.: 50 PRINTUSING 3, X, SIN(X), CØS(X)
CHA 50/3/1/ führt zu:
50 PRINTUSING 1, X, SIN(X), CØS(X)
CHA 50/X/A/* führt zu:
50 PRINTUSING 1, A, SIN(A), CØS(A)

Eine Reihe von System-Befehlen sind gleich denen in Mark I , darunter:
BAS BYE GØØ HEL LIS NEW ØLD REN REP SCR SYS TAP.

8.9 Verschiedenes

Programmnamen können, beginnend mit einem Buchstaben, bis zu acht Zeichen lang sein. (Die Verwendung anderer Zeichen als Buchstaben oder Ziffern sollte vermieden werden.)

Normale BASIC-Programme sind sequentiell zugreifbar im ASCII-Code gespeichert. Für BASIC-Dateien gilt:

1 Wort = 36 Bits = 4 Zeichen
1 Zeile (1 record) = (max.) 39 + 1/2 Worte = 158 Zeichen
1 PSU (Program Storage Unit; 1 Seite) = 1260 Zeichen

Nach Einleitung der Trennung vom Computer (durch z.B. GØØ) erfolgt eine Ausgabe der folgenden Form:

0001.15 CRU 0000.11 TCH 0002.37 KC
ØFF AT 18:40 12/05/73

Es bedeuten dabei:

CRU ... Computer Resource Units
TCH ... Terminal Connect Hours
KC ... Kilo-Characters transmitted

Mark I-Programme sind in Mark III ausführbar und zwar unter folgenden Bedingungen:

(1) Die Anweisung END (größte Zeilennummer) ist vorhanden.
(2) :-Anweisungen wurden, wenn nötig, den neuen Regeln angepaßt.
(3) So im Programm verwendet, sind Rechner-Unendlich (größte darstellbare Zahl) und Rechner-Auflösungsvermögen (kleinste darstellbare Zahl ungleich Null) adjustiert.
(4) RND(−1) ist auf RAN geändert (eigene Zeile) und RND (0) ist durch RND ersetzt.
(5) Das Programm enthält keine FILES-Anweisungen.

8.10 Demonstration des Ablaufs eines Programms in Mark III

```
SYS BAS

READY
NEW RAT

READY
TAP
READY
100'REIHE FUER LØG10(X)
110READA1,A3
120DATA.86304,.36415
130PRINT"A1="A1,"A3="A3\PRINT
140PRINT"  X","  T","  Y","  E"
150READX
160DATA.5,1,2
170Y=A1*(T=(X-1)/(X+1))+A3*T↑#
180L=.43429448Ø*LØG(X)
190PRINTX,T,Y,(Y-L)/L\GØTØ150
999END
SAV

READY
RUN-1


RAT          12:39        10/28/74

ILLEGAL FØRMULA IN 170
```

```
USED      .05 UNITS
CHA 170/3/# DELETED
CHA 170/#/3/

READY
REP

READY
RUN-1


RAT          12:40        10/28/74

A1= 0.86304     A3= 0.36415

  X              T               Y               E
 0.5            -0.333333       -0.301167        4.55245E-4
 1               0               0
DIVISI0N BY ZER0 IN 190
 1.70141E+38
 2              0.333333         0.301167        4.55245E-4
0UT 0F DATA IN 150


USED      .12 UNITS
PUR RAT;RAE

READY
G00
0003.37  CRU     0000.27  TCH     0005.92  KC

0FF AT 12:41     10/28/74
```

9. Beispiele (Mark III)

9.1 Demonstration der PRINTUSING-Anweisung und der Zeilenmaske

Dieses Beispiel demonstriert die Wirkung der alphamerischen Schablonen L, R, C und E, und neue Steuerungsmöglichkeiten bei numerischen Schablonen.

```
RUN-1


RAPU          10:15          07/08/74

*A   **   A** A **A *
*AB  **  AB**AB  **AB*
*ABC**ABC**ABC**ABC*
*ABC**ABC**ABC**ABCD*

    12@@+   12@@   +12
   -12@@-   12@@   -12
   12.35@@+ 12.35@@ +12.35
  123.46E-01@@+123.46E-01@@+123.46E-01


USED       .15 UNITS


LIST


RAPU          10:16          07/08/74

1:*'LL**'RR**'CC**'E*
2:######@@+#####@@++++#
3:####.##@@+###.##@@++++.##
4:####.##↑↑↑↑@@+###.##↑↑↑↑@@++++.##↑↑↑↑
10A$="A"
20B$="AB"
30C$="ABC"
40D$="ABCD"
50PRINTUSING1,A$,A$,A$,A$
60PRINTUSING1,B$,B$,B$,B$
70PRINTUSING1,C$,C$,C$,C$
80PRINTUSING1,D$,D$,D$,D$
90PRINT
100F=12.3456
```

```
110PRINTUSING2,F,F,F
120PRINTUSING2,-F,-F,-F
130PRINTUSING3,F,F,F
140PRINTUSING4,F,F,F
999END

READY
```

Die eingestreuten *-Zeichen bei den alphamerischen Schablonen, bzw. @-Zeichen bei den numerischen Schablonen, dienen zur Hervorhebung der Schablone-bedingten Leerstellen.

Es soll an dieser Stelle darauf hingewiesen werden, daß die Verwendung von mehr als 1+8 # bei einfacher Genauigkeit, bzw. 1+19 # bei erhöhter Genauigkeit, zu Fehlausgaben führen kann.

9.2 Demonstration der MARGIN-Anweisung

Dieses Beispiel demonstriert die Wirkung der MARGIN-Anweisung. Die ersten beiden Zeilen dienen nur zur Orientierung bezüglich der Druckposition.

```
NEW RAMA

READY
TAP
READY
10 PRINT"0          1          2"
20 PRINT"01234567890123456789O"
30 PRINT"0HNE MARGIN"
40 I=I+1
50 IF I=1 THEN 80
60 MARGIN 15
70 PRINT"MIT MARGIN 15"
80 PRINT TAB(12) "A" "B" "C" "D" "E" "F"
90 PRINT TAB(12) "ABCDEF"
100 PRINT "A","B"
110 PRINT TAB(12);1;2;3;4;5;6
120 PRINT TAB(12);123456
130 IF I=1 THEN 40
999 END
```

```
RUN-1

RAMA          08:24          06/15/74

0          1          2
012345678901234567890
OHNE MARGIN
               ABCDEF
               ABCDEF
A                 B
                   1   2   3   4   5   6
                   123456
MIT MARGIN 15
               ABC
DEF

ABCDEF
A
B
                   1
  2   3   4   5   6

  123456


USED       .12 UNITS
```

9.3 n-maliges Werfen eines m-seitigen Würfels

(RAN; RND; Addition alphamerischer Größen)

Dieses Beispiel demonstriert die Verwendung von RAN und RND, sowohl als auch die "Addition" alphamerischer Größen.

Ein m-seitiger "Würfel" soll n-mal geworfen werden. Die Häufigkeit des Auftretens jeder Seite soll in Form eines "Säulendiagramms" dargestellt werden.

```
NEW RADICE


READY
TAP
READY
10RAN
20READM,N
30DATA10,100,10,100
40PRINTM"SEITEN   "N"WUERFE"
50PRINT
60FORI=1TON
70K=M*RND+1
80TS(K)=TS(K)+"*"
```

```
90NEXTI
100FØRI=1TØM
110PRINTTS(I)
120TS(I)=""
130NEXTI
140PRINT
150GØTØ20
999END


RUN-1


RADICE       07:46        10/13/73

  10 SEITEN    100 WUERFE

*********
********
********
**********
*********
******
******
**********
*******
**********

  10 SEITEN    100 WUERFE

*******
**********
****
*******
**********
*********
********
********
**********
**********

ØUT ØF DATA IN 20


USED       .33 UNITS
```

Besprechung:
10...initialisiert den Zufallszahlengenerator mit einer Zufallszahl
20,30,40...es ist keine Absicherung gegen Eingabe "unmöglicher" *m*- und/oder *n*-Werte vorgesehen

70...eine Zahl k ($1 < k < m + 1$) wird erzeugt. Der ganzzahlige Teil von k ist die bei diesem Wurf aufscheinende Punktezahl. INT(K) ist nicht notwendig, da die folgende Indizierung T$(K) wie INT wirkt
80... die "Säule" der "K"-Punkte-Fälle wird um einen * erhöht
110...Ausdrucken der "Säulen"
120...die eben gedruckte Säule wird auf "leer" gestellt in Vorbereitung für die nächste Wurfserie

Anmerkung: Das Programm prüft die „Gleichverteilung" der durch RND erzeugten Zufallszahlen.

Ohne Verwendung von alphamerischen Variablen wird das Programm ein wenig länger, läuft aber doppelt so schnell ab.

9.4 Zuordnung von Normwerten
(Mehrzeiliges DEF mit Wertzuweisung)

Dieses Beispiel demonstriert die Verwendung des mehrzeiligen DEF mit Wertzuteilung.

```
LIST


RANORM        09:35          06/15/74

10'UNTERPRØGRAMM
20 DEF FNN(R) P,W
30 P=10↑INT(.434294482*LØG(R))
40 R=R/P
50 W=W+1
60 IF R>W(W+1) THEN 50
70 FNN=W(W+INT(R/W(W)*R/W(W+1)))*P
80 FNEND
200'RUFENDES PRØGRAMM
210 DIM W(25)
220 FØR I=1 TØ 25
230 READ W(I)
240 NEXT I
250 DATA1,1.1,1.2,1.3,1.5,1.6,1.8,2,2.2,2.4,2.7,3,3.3
260 DATA3.6,3.9,4.3,4.7,5.1,5.6,6.2,6.8,7.5,8.2,9.1,10
270 PRINT "ZUØRDNUNG VØN NØRMWERTEN"
280 PRINT "FUER ERRECHNETE WERTE"
290 PRINT
```

```
300 PRINT "RECHENWERT","NØRMWERT"
310 RAN
320 FØR I=1 TØ 10
330 X=(10*RND+1)↑(7*RND+1)
340 PRINT X,FNN(X)
350 NEXT I
999 END


READY


RUN-1

RANØRM        09:37         06/15/74

ZUØRDNUNG VØN NØRMWERTEN
FUER ERRECHNETE WERTE

RECHENWERT       NØRMWERT
 20164.9          20000
 2140.33          2200
 15.8514          16
 31.3717          30
 11.5248          12
 13687.8          13000
 3.29405E+7       33000000
 13.644           13
 76.2233          75
 2734.72          2700


USED      .17 UNITS
```

Besprechung:

20...die Scheinvariable in diesem Beispiel hat unterschiedlichen Namen im rufenden Programm (340,X) und im Unterprogramm (20,R). Die Liste der Lokalvariablen (zweite Variablenliste) mit P und W könnte hier weggelassen werden, da P und W im rufenden Programm ohnehin nicht aufscheinen. Die Anführung von P und W gestattet jedoch die Verwendung des Unterprogramms mit einem beliebigen rufenden Programm. Voraussetzung ist nur, daß der Name des Feldes für die Normwerte W( ) ist (wegen des globalen Charakters indizierter Variablen)

70...der vom rufenden Programm verlangte Funktionswert wird hier erzeugt

80...beschließt das Unterprogramm FNN. FNEND hat die Bedeutung von: "Setze nun mit der Ausführung des rufenden Programms fort".

310...startet den Zufallszahlengenerator mit einer Zufallszahl

330...die beiden RND erzeugen je eine Zufallszahl

340...FNN(X) in dieser Zeile bringt das Unterprogramm FNN, unter Verwendung des z.Z. gültigen Wertes von X zum Ablaufen.

9.5 Sortieren numerischer Daten (mehrzeiliges DEF ohne Wertzuweisung)

Dieses Beispiel demonstriert die Verwendung des mehrzeiligen DEF ohne Wertzuteilung; es sortiert n Werte in steigender Folge.

```
NEW RASØRT

READY
TAP
READY
99 'SØRTIER-UNTERPRØGRAMM
100 DEF FNS(N)I,J,K,T
110 FØR I=1 TØ N-1
120 T=X(I)
130 K=I
140 FØR J=I+1 TØ N
150 IF X(J)>=T THEN 180
160 T=X(J)
170 K=J
180 NEXT J
190 X(K)=X(I)
200 X(I)=T
210 NEXT I
220 FNEND
299'HAUPTPRØGRAMM
300 DIM X(20)
310 N=12
320 PRINT "UNSØRTIERT"
330 FØR I=1 TØ N
340 X(I)=200*RND-100
350 PRINT X(I);
360 NEXT I
370 PRINT
380 PRINT "SØRTIERT"
390 T=FNS(N)
400 FØR I=1 TØ N
410 PRINT X(I);
420 NEXT I
430 PRINT
999 END

RUN-1

RASØRT      07:44        03/27/73

UNSORTIERT
 96.7701 -27.5452 -49.8931 -32.3852 -49.9981 -31.5389  35.3474  64.0033
-41.9336  36.638 -25.2954  70.7559
SØRTIERT
-49.9981 -49.8931 -41.9336 -32.3852 -31.5389 -27.5452 -25.2954  35.3474
 36.638   64.0033  70.7559  96.7701

USED      .14 UNITS
```

Besprechung:
99...Kommentar
100...Die Scheinvariable N in diesem Beispiel hat gleichen Namen im Unterprogramm und Hauptprogramm(390)
110...210...Sortier-Schleife
120,150,160,200...wegen der Anführung von T in 100 als Lokalvariable ist T in diesen Zeilen nicht identisch mit T in 390
120...X(I) als Globalvariable ist identisch mit X(I) im Hauptprogramm (340). I jedoch ist, wegen der Anführung in 100, im Haupt- und Unter-Programm Programm nicht identisch.
220...schließt das Unterprogramm FNS ab
299...Kommentar
300...gilt, da X( ) global ist, für Haupt- und Unterprogramm
330...360...Belegung (und Ausdrucken) des Feldes X
340...Erzeugung von Zahlen zwischen −100 und +100.
390...diese Anweisung bringt das Unterprogramm FNS zum Ablaufen. Da in diesem Beispiel kein Funktionswert erzeugt wird, es wird ja nur umgruppiert, fehlt im Unterprogramm die Zeile mit FNS=...
400...420...Ausdrucken der sortierten Werte
999...beschließt das gesamte Programm

An RASØRT soll auch die Anwendung implizierter Zuordnungen,sowie die „gepackte" Programmschreibweise gezeigt werden.

```
0'SØRTIER-UNTERPRØGRAMM
1DEFFNS(N)I,J,K,T\FØRI=1TØN-1\T=X(K=I)\FØRJ=I+1TØN\IFX(J)>=TTHEN2\T=X(K=J
2NEXTJ\X(K)=X(I)\X(I)=T\NEXTI\FNEND
299'HAUPTPRØGRAMM
300DIMX(20)\N=12\PRINT"UNSØRTIERT"\FØRI=1TØN\PRINTX(I)=200*RND-100;\NEXT
400PRINT\PRINT"SØRTIERT"\T=FNS(N)\FØRI=1TØN\PRINTX(I);\NEXTI\PRINT\END
```

Da 158 Zeichen pro Programmzeile zulässig sind, am Fernschreiber aber nur 72 Positionen bedruckt werden können, kann es zum Überdrucken in der letzten Druckposition der Zeile kommen.

Das Ende der Zeile 1 lautet ... T=X(K=J)
Das Ende der Zeile 300 lautet ... NEXTI

Eine derartige "Packung" erzeugt unübersichtliche Verhältnisse und ist daher bei der Entwicklung eines Programms eher hinderlich. Verdichtung spart aber Speicherplatz und Laufzeit, und sollte daher bei durchgetesteten, abgeschlossenen Programmen angewendet werden.

9.6 Lösung eines linearen Gleichungssystems mit Matrizenoperationen (MAT-Anweisungen)

Dieses Beispiel demonstriert die Verwendung einer Reihe von MAT-Befehlen an Hand der Auflösung eines linearen Gleichungssystems durch Matrix-Inversion.

```
NEW KAMA1


READY
)AP
READY

)'LØESUNG FINES LINEAREN GLEICHUNGSSYSTEMS MIT MATRIZEN
2'MAX. GRØESZE: 20 X 20
3DIM A(440),C(41),I(440),E(440)
4)##.##########  ##.##########  ##.##########  ##.##########  ##.##########
9N=4
10DATA2.38622,1.9524,-3.27441,1.58376
20DATA3.21202,-.8644,2.42487,-3.20328
30DATA1.44144,2.95255,-2.14359,1.86444
40DATA4.171122,3.62082,-1.68712,-2.26148
50DATA2.16386,3.28309,1.42215,5.21003
100MATREADA(N,N)
110PRINT"MATRIX DER KØEFF.:";
120MATPRINTUSING4,A
130PRINT
140MAT I=ZER(N,N)
150MAT I=INV(A)
160PRINTUSING"DET(KØEFF.)=##.##########",DET
170PRINT
180MATREADC(N,1)
190PRINT"MATRIX DER KST.:";
200MATPRINTUSING4,C
210PRINT
220PRINT"F=A*I";
230MAT F=ZER(N,N)
240MAT F=A*I
250MATPRINTUSING4,E
260PRINT
270PRINT"LØESUNGSMATRIX:";
280MAT F=ZER(N,1)
290MAT F=I*C
300MAT I=ZER(1,N)
310MAT I=TRN(E)
320MATPRINTUSING4,I
330PRINT
340PRINT"KONTRØLLE: KST=KØEFF.*LØESG.";
350MAT C=A*F
360MATPRINTUSING4,C
370END


RAMA1          12:02          12/14/73

MATRIX DER KØEFF.:
 2.38622E+00   1.95240E+00  -3.27441E+00   1.58376E+00
 3.21202E+00  -8.64400E-01   2.42487E+00  -3.20328E+00
 1.44144E+00   2.95255E+00  -2.14359E+00   1.86444E+00
 4.17112E+00   3.62082E+00  -1.68712E+00  -2.26148E+00

DET(KØEFF.)= 8.05236E+01

MATRIX DER KST.:
 2.16386E+00
 3.28309E+00
 1.42215E+00
 5.21003E+00
```

```
F=A*I
 1.00000E+00   6.29525E-09   3.23570E-08  -5.07137E-08
-3.75212E-08   1.00000E+00   3.12183E-08  -3.93245E-09
-2.18112E-09  -4.12933E-09   1.00000E+00  -3.91104E-08
-1.52562E-08  -1.61475E-08   3.06386E-08   1.00000E+00

LØESUNGSMATRIX:
 8.08186E-01   2.37663E-01  -1.02462E-01  -3.56220E-01

KØNTRØLLE: KST=KØFFF.*LØESG.
 2.16386E+00
 3.28309E+00
 1.42215E+00
 5.21003E+00


USED      .28 UNITS
```

Besprechung:

3...in der DIM-Anweisung kommt es nicht auf die tatsächliche Anzahl der Zeilen und Spalten an, sondern auf die Reservierung von ausreichendem Speicherplatz. Bei maximalen Indizes von, z.B., I=20 und J=20 werden (20+1) . (20+1) = 441 Speicherplätze reserviert. Diese Reservierung kann über DIM A (20,20) oder aber auch über DIM A (440) erfolgen. Letztere Version hat den Vorteil, daß man bei der Wahl der tatsächlichen Zahl der Zeilen und Spalten flexibler sein kann als bei DIM A (20,20)[1]. Die Speicherplätze mit Index Null (A(n,0); A(0,n)) werden zwar reserviert, aber von MAT-Anweisungen nicht verwendet.

9...die tatsächliche Ordnung des Gleichungssystems wird angegeben

10,20,30,40,50...die Daten müssen mit der in 9 gegebenen Ordnung übereinstimmen. Eingelesen wird zeilenweise.

100...die Koeffizientenmatrix wird eingelesen und gleichzeitig A auf 4x4 dimensioniert

140...vor jeder Matrix-Operation muß dem Compiler die genaue benötigte Anzahl von Zeilen und Spalten angegeben werden. Dies kann über MAT READ, MAT ZER, MAT CØN, MAT IDN erfolgen.

160...der Wert der Determinante der Koeffizientenmatrix wird ausgegeben. Er kann bei der Erkennung nahezu singulärer Matrizen helfen.

240...E=A*I (Originalmatrix mal ihre inverse Matrix) gibt (theoretisch) die Einheitsmatrix. Werte von 10^{-8} bis 10^{-9} ober- und unterhalb der Hauptdiagonale zeigen die Genauigkeit des verwendeten Computers an (im gegebenen Fall 8 Stellen). Auch diese Operation kann zur Kontrolle, ob es ich um eine nahezu singuläre Matrix handelt dienen.

---

[1] Werden einzelne Elemente einer Matrix angesprochen, z.B. durch V=A(N,2) oder A(K,L) = LØG(X), dann muß die Zweidimensionalität dem Compiler signalisiert werden. Das kann einfach durch DIM A(440,0) geschehen.

310...eine rein "kosmetische" Operation; sie gibt die Lösung als Zeilenvektor, statt als Spaltenvektor
350...endgültige Kontrolle der Gültigkeit der Lösungen

9.7 Numerisches Differenzieren (Zwei-Punkte-Methode) (DØUBLE PRECISIØN; alphamerische PRINTUSING-Masken)

Dieses Beispiel demonstriert die Anwendung von DØUBLE PRECISIØN und die Manipulation von alphamerischen Größen im Zusammenhang mit einer Zeilenmaske für PRINTUSING.

Dieses Beispiel zeigt auch an Hand einer einfachen numerischen Differentiation wie durch den Verlust signifikanter Stellen bei Subtraktion die Genauigkeit sinkt. Das hat zur Folge, daß bei $\Delta x \rightarrow 0$ eine schlechtere, statt einen besseren Näherung für den Differentialquotienten erhalten wird.

```
NEW RADYDX

READY
TAP
READY
 0 DØUBLE PRECISIØN
 2 M$="##.##################"
 4 H$="X="+M$+"   Y="+M$
 6 L$="#.##################   "+M$+"   "+M$
10 DEF FNY(X)=EXP(X)
20 DEF FND(X)=(FNY(X+D)-FNY(X-D))/2/D
30 READ X
31 DATA 1
40 PRINTUSING H$,X,FNY(X)
50 D=10
60 PRINT
70 PRINT TAB(10)"D"TAB(31)"DY/DX"TAB(52)"ABS.FEHLER"
80 FØR I=1TØ10
90 D=D/100
100 A=FND(X)
110 PRINTUSING L$,D,A,ABS(2.7182818284590452 35-A)
120 NEXT I
130 GØTØ 30
999 END
```

```
RUN-1

RADYDX        09:00        06/15/74

X= 1.000000000000000000  Y= 2.718281828459045235

          D                    DY/DX                  ABS.FEHLER
 .1000000000000000000   2.722814563947417237    .004532735488372003
 .0010000000000000000   2.718282281506038416    .000000453046993182
 .0000100000000000000   2.718281828504292357    .000000000045247123
 .0000001000000000000   2.718281828457831922    .000000000001213312
 .0000000010000000000   2.718281828188712614    .000000000270332620
 .0000000000100000000   2.718281832671210971    .000000004212165737
 .0000000000001000000   2.718281261432191343    .000000567026853891
 .0000000000000010000   2.717837348405194310    .000444480053850924
 .0000000000000000100   2.715587628897286888    .002694199561758346
 .0000000000000000001   -.745388993583784299   3.463670822042829533
OUT OF DATA IN 30

USED      .32 UNITS
```

Besprechung:

0...Anweisung mit der niedrigsten Zeilennummer (nur ' und DIM sind vor DØUBLE PRECISIØN zulässig)

2,4,6...da die 19-stellige Fixkomma-Schablone mehrmals gebraucht wird ist es platzsparend sie nur einmal zu definieren (2),und dann entsprechend mit anderen Schablonen zu kombinieren (4 und 6)

10...definiert eine Funktion, hier e^x

20...definiert die erste Ableitung einer Funktion über den Anstieg der Sehne zwischen zwei symmetrisch zum betrachteten Punkt liegende Punkte.

40...x und y=f(x) wird ausgedruckt

80...120...in der Schleife wird mit immer kleineren Schritten der Anstieg der Sehne berechnet und mit der tatsächlichen ersten Ableitung verglichen

Die Ausgabe zeigt, daß die Genauigkeit erst steigt, ein Maximum erreicht, und dann wieder sinkt. Grund dafür ist folgender: In der vierten Zeile der Ausgabe (z.B.) gehen bei der Subtraktion in FND(X) (20) sieben signifikante Stellen verloren, d.h. die Genauigkeit kann nur mehr in der Größenordnung von 19 – 7 = 12 Stellen sein. Bei kleineren Werten von D gehen noch mehr signifikante Stellen verloren, da ja die y-Werte immer näher aneinanderrücken, d.h. immer mehr Stellen übereinstimmen.

Allgemein kann zu DØUBLE PRECISIØN folgendes gesagt werden: Die praktische Genauigkeit – einschließlich Runden im Gleitkomma-Format – ist 16 Stellen. Die eingebauten Funktionen geben Werte genau auf etwa 18 Stellen (SQR, EXP, LØG sind genauer als SIN, CØS, TAN, CØT; ziemlich ungenau ist ATN)[1]

[1] Im normalen (single precision) Modus beträgt die Genauigkeit sämtliche 8 Stellen (d.h. die Funktionen werden im DP-Modus gerechnet).

9.8 Integration mittels Simpson-Regel
(Vollständige Durchentwicklung eines Programms)

Dieses Programm soll, beginnend mit der Problemstellung, vollkommen durchbesprochen werden.

Problemstellung:
Es ist ein mehrzeiliges Funktionsunterprogramm zu schreiben, welches das Integral einer gegebenen Funktion mittels der Simpson-Regel findet. Integrierbarkeit der Funktion sei vorausgesetzt.

Gegeben:	Die zu integrierende Funktion über DEF FNY(X)=..., sowie Anfangswert, Endwert und gewünschte Genauigkeit.
Gesucht:	Das Unterprogramm in dem der Anfangswert, der Endwert und die Genauigkeit als Scheinvariablen, etwaige andere Variablen als Lokalvariablen deklariert sind. Weiters ist ein Hauptprogramm zum Testen dieses Unterprogramms zu schreiben.
Ausgabe:	Integrationsgrenzen, Integralwert, Genauigkeit

Problemlösung:
1) Definition des Problems
   Durch die Problemstellung gegeben. Grenzwerte = Computergrenzwerte
2) Bestimmung der Methode
   Durch die Problemstellung ist die Simpson-Regel gegeben. Andere Möglichkeiten wären: Trapez-Regel, Romberg-Regel, Gauß-Regel.
3) Mechanisierung der Methode
   Die Simpson-Regel für drei Punkte, d.h. 2 Intervalle, lautet:

$$F_1 = \frac{h_2}{3} \cdot (y_b + 4 \cdot y_m + y_e)$$

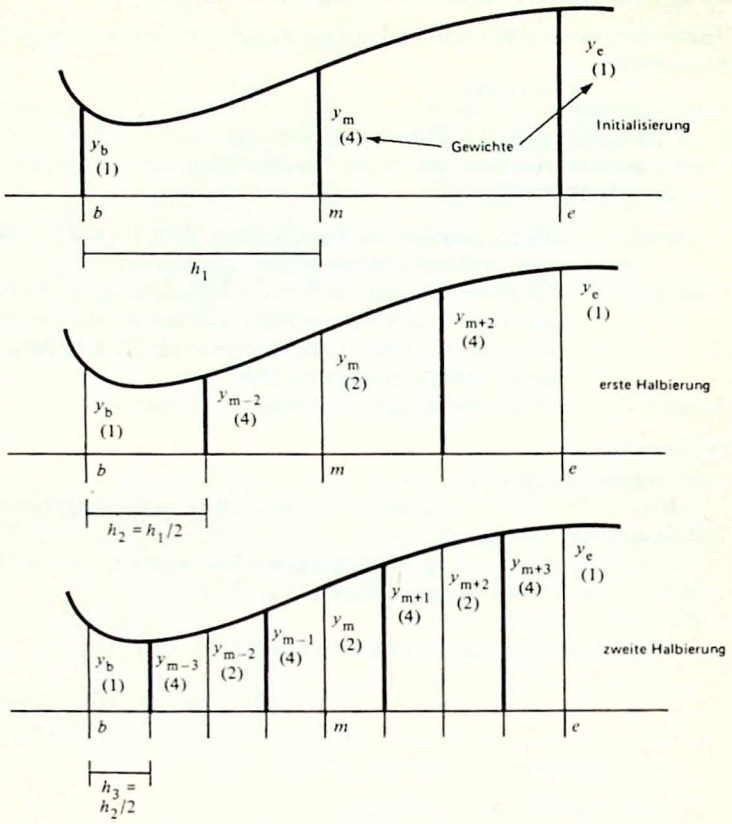
wobei b die untere Grenze (Beginn), e die obere Grenze (Ende) und m die Intervallmitte bezeichnet.

Für fünf Punkte (oder 4 Intervalle):

$$F_2 = \frac{h_2}{3} \cdot (y_b + 4 \cdot y_{m-2} + 2 \cdot y_m + 4 \cdot y_{m+2} + y_e)$$

Von Vorteil, besonders bei "Binär-Maschinen" (Computer die im Dual-System rechnen), ist "Intervallhalbierung", d.h. $h_2 = h_1/2$. Einmal berechnete y-Werte werden natürlich bei weiteren Halbierungen weiterverwendet. Die "Halbierungen" werden solange fortgesetzt, bis zwei aufeinanderfolgende "Flächenwerte" innerhalb der geforderten Genauigkeit gleich sind.

Graphische Darstellung der Intervallhalbierung ("neue" y-Werte stark ausgezogen):



Die Betrachtung der Zeichnung zeigt: Die Endwerte haben unverändert das Gewicht "1"; die "neuen" Werte haben das Gewicht "4"; die "alten" Werte haben das Gewicht "2". Die mit einem Drittel des Schrittes multiplizierte Summe der "gewichteten" y-Werte gibt die Fläche, den Integralwert.

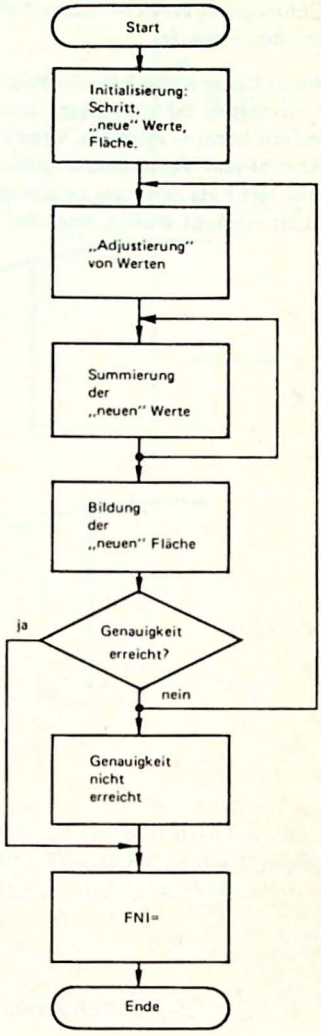
Daraus ergibt sich folgender Auswertungsmechanismus:
Starte mit einem Schritt gleich der Hälfte des Integrationsintervalls. Initialisiere die "neuen", mit $4 \cdot h/3$ zu multiplizierenden y-Werte mit: $r = 4\,h/3 \cdot y_m$. Initialisiere die Fläche mit: $f = r + h/3 \cdot (y_b + y_e)$
Nun beginnt eine Schleife:
Der Schritt wird halbiert.
Die Hälfte der "neuen" Werte (r) von vorhin müssen von der Fläche (f) abgezogen werden (Gewicht jetzt nur mehr 2). Der resultierende Wert muß dann noch halbiert werden (wegen $h \rightarrow h/2$).

Nun Berechnung der "neuen" y-Werte (Schritt $2 \cdot h$); Multiplikation mit $4 \cdot h/3$; Addition zur "adjustierten" Fläche von vorhin.
Vergleich der jetzt erhaltenen Fläche mit der vorher erhaltenen; ist die gewünschte Genauigkeit erreicht, dann Aussprung aus der Schleife, sonst neuerliches Durchlaufen der Schleife.

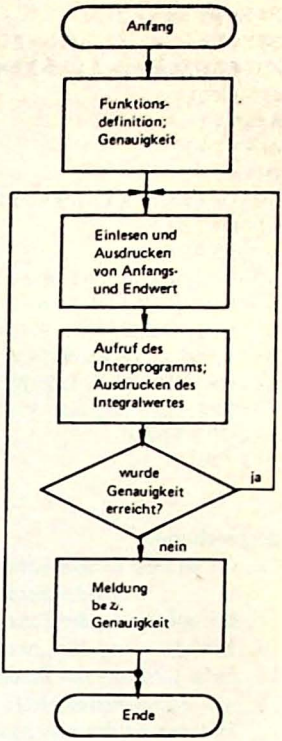
Wenn die gewünschte Genauigkeit nicht erreicht werden kann – sei es durch schlechte Konvergenz oder weil eine größere Genauigkeit gefordert wurde als sie das Verfahren liefern kann – muß zur Vermeidung einer endlosen Schleife nach einer bestimmten Anzahl von Versuchen abgebrochen werden. Wie bei der Iteration ist es wenig sinnvoll mehr als zehnmal zu halbieren. Die Tatsache, daß die gewünschte Genauigkeit nicht erreicht wurde, muß dem Hauptprogramm gemeldet werden.

4) Flußdiagramm

Flußdiagramm des Unterprogramms:



Flußdiagramm des Hauptprogramms:



5) Codieren

```
0'SIMPSØN MIT INTERVALL-HALBIERUNG
1:GRENZEN: ##.########↑↑↑↑ ... ##.########↑↑↑↑
2:INTEGRALWERT: ##.##########↑↑↑↑  GENAUIGKEIT: ##.↑↑↑↑
10DEFFNI(B,E,T)F,H,K,L,R,S
12G9=0
14H=(E-B)/2
16R=4*FNY(B+H)*H/3
18F=R+(FNY(B)+FNY(E))*H/3
20FØRL=2TØ7
22H=H/2
24S=F
26F=F/2+R/4
28R=FNY(B+H)
30FØRK=3TØ2↑L STEP2
32R=R+FNY(B+K*H)
```

```
34NEXTK
36R=4*R*H/3
38F=F+R
40IFABS(S/F-1)<=T1THEN46
42NEXTL
44G9=1
46FNI=F
48FNEND
500DEFFNY(X)=COS(SIN(X)-X)
510G=1E-6
520READA,E
530DATA0,3.14159265
540PRINTUSING1,A,E
550PRINTUSING2,FNI(A,E,G),G
560IFG9=01HEN580
570PRINT"SCHLECHTE KONVERGENZ. GENAUIGKEIT NICHT GARANTIERT.
580PRINT
590GOTO520
999END
```

Besprechung:
12...G9 ist eine Globalvariable die zur Anzeige, ob die gewünschte Genauigkeit erreicht wurde dient
14...Initialisierung der Schrittgröße
16...Initialisierung der „neuen" y-Werte
18...Initialisierung der Flächen, des Integralwertes
20...42...Sicherheitsschleife
22...Halbierung des Schrittes
24..."alte" Fläche = "neue " Fläche
26..."Adjustierung" der bisherigen Fläche
28...Initialisierung der "neuen" y-Werte
30...34...Summation der "neuen" Werte
36...Multiplikation der "neuen" Werte mit $4 \cdot h/3$
38...Bildung der "neuen" Fläche
40...Prüfung der Genauigkeit
44...G9=1, wenn die Genauigkeit nach 7 Halbierungen nicht erreicht wurde
46...das Ergebnis wird dem Hauptprogramm zur Verfügung gestellt

6) und 7) Testen

```
TAP
READY
500DEFFNY(X)=COS(SIN(X)-X)
530DATA0,3.14159265
RUN-1
```

```
RASIMP        13:08          10/16/74

GRENZEN:   .0000000E+00 ...   3.1415927E+00
INTEGRALWERT: 1.3824597E+00   GENAUIGKEIT: 1.E-06

USED       .26 UNITS
TAP
READY
500DEFFNY(X)=SIN(X)
530DATA0,6.28318531
RUN-1

RASIMP        13:09          10/16/74

GRENZEN:   .0000000E+00 ...   6.2831853E+00
INTEGRALWERT:-1.4111509E-09   GENAUIGKEIT: 1.E-06
SCHLECHTE KØNVERGENZ. GENAUIGKEIT NICHT GARANTIERT.

ØUT ØF DATA IN 520

USED       .32 UNITS
TAP
READY
500DEFFNY(X)=EXP(-X↑2)
530DATA0,.5
RUN-1

RASIMP        13:10          10/16/74

GRENZEN:   .0000000E+00 ...   5.0000000E-01
INTEGRALWERT: 4.6128103E-01   GENAUIGKEIT: 1.E-06

ØUT ØF DATA IN 520


USED       .24 UNITS
TAP
READY
500DEFFNY(X)=SIN(X)/X
530DATA.5,1
RUN-1

RASIMP        13:10          10/16/74

GRENZEN: 5.0000000E-01 ...   1.0000000E+00
INTEGRALWERT: 4.5297566E-01   GENAUIGKEIT: 1.E-06

ØUT ØF DATA IN 520

USED       .24 UNITS
```

Das Integrierprogramm soll für weitere Anwendungen "gepackt" werden. Folgend eine gepackte Version (Hauptprogramm nicht gepackt). Der in den Zeilen 4 und 6 in der letzten Druckposition vielfach überdruckte Text kann aus der ungepackten Version ersehen werden.

```
4DEFFNI(B,E,T)F,H,K,L,R,S\G9=0\F=(R=4*FNY(B+(H=(E-B)/2))*H/3)+(FNY(B)+F█
6FØRK=3TØ2↑LSTEP2\R=R+FNY(B+K*H)\NEXTK\F=F+(R=4*R*H/3)\IFABS(S/F-1)<=TT█
8FNI=F\FNEND
```

9.9 Zeichnen von sechs Funktionen am Terminal
(REP$; SCN; Arbeiten mit alphamerischen Größen)

Dieses Beispiel demonstriert die Anwendung von REP$ und SCN – Anweisungen die an alphamerischen Größen operieren – sowie die Anwendung von variablen Zeilenmasken in PRINTUSING.

```
NEW KAPL016

READY
TAP
READY
1:DEMONSTRATIONS-BEISPIEL
2:0 - X↑6    * - X↑5    + - X↑4    : - X↑3    = - X↑2    . - X↑1
3:           ↑....↑....↑....↑....↑....↑....↑....↑....↑....↑....↑....↑....↑....↑
10DEFFNA(X)=X↑6
20DEFFNB(X)=X↑5
30DEFFNC(X)=X↑4
40DEFFND(X)=X↑3
50DEFFNE(X)=X↑2
60DEFFNF(X)=X
100READN,A,E,D
110DATA6,-1.5,1.5,.125
120FØRI=0TØN-1
130READS(I),Z(I),P$(I)
135DATA20,40,"0",20,40,*,20,40,"+",20,40,:,20,40,=,20,40,"."
140NEXTI
150PRINTUSING1
160PRINTUSING2
170PRINT
180PRINTUSING3
190FØRT=ATØESTEPD
200ØNNGØTØ260,250,240,230,220,210
210F(5)=FNF(T)
220F(4)=FNE(T)
230F(3)=FND(T)
240F(2)=FNC(T)
250F(1)=FNB(T)
260F(0)=FNA(T)
270L$=""
280F=0
290FØRI=0TØN-1
300P=INT[S(I)*F(I)+Z(I)-8.5]
```

```
310 IF ABS(P-31)>30 THEN 350
320 IF F>=P THEN 335
330 F=P
335 IF SCN(L$," ",1,P)<>P THEN 350
340 L$=REP$(L$," ",P$(I),1,P)
350 NEXT I
360 L$="##.##↑↑↑↑-"+REP$(L$," ","",-1,F+1)
370 PRINT USING L$,T
380 NEXT T
999 END
```

Besprechung:

270...die „Arbeitszeile" wird mit 61 Balnks initialisiert

280...die größte mit einem Drucksymbol belegte Position in einer Arbeitszeile wird initialisiert

290...350...Schleife zum Einsetzen der Drucksymbole in die Arbeitszeile

300...Bestimmung der Position des Drucksymbols für die I-te Funktion (Rangordnung der Funktionen beim Drucken rein durch Reihenfolg bestimmt)

310...Ausscheidung von Positionen < 1 und > 61

320,330...die größte mit einem Drucksymbol belegte Position in der Arbeitszeile wird gesetzt

335...es wird geprüft, ob die in 300 errechnete Position schon mit einem Drucksym bol belegt ist

340...ist die P-te Position noch unbelegt, wird das Drucksymbol für die I-te Funkti eingesetzt.

360...die Zeilenmaske wird vorbereitet; REP$ in dieser Programmzeile unterdrückt Blanks rechts der größten mit einem Drucksymbol belegten Positio

370...Ausdrucken der Zeile

Es gibt natürlich noch einige andere Lösungsvarianten. Die um eine Zeile kürzere Lösungsvariante

```
360 PRINTUSING "# # . # # ↑ ↑ ↑ ↑ -' E", T,REP$(L$," ","", -1,F+1)
370 NEXT T
```

führt allerdings bei bestimmten Datenkombinationen zu einer Lauffehlermeldung.

```
RUN-1

RAPLØT6      11:13        01/02/74

DEMONSTRATIONS-BEISPIEL
0 - X↑6   * - X↑5   + - X↑4   : - X↑3   = - X↑2   . - X↑1

          ↑....↑....↑....↑....↑....↑....↑....↑....↑....↑....↑....↑....↑
-1.50E+00-.
-1.37E+00-   .
-1.25E+00-     .
-1.12E+00-  :     .                                                  =
-1.00E+00-          *                                          0
-8.75E-01-             .   :  *                    0  +  =
-7.50E-01-               .      :  *         0 +     =
-6.25E-01-                  .      :  *  0 +     =
-5.00E-01-                     .      :*0+    =
-3.75E-01-                        .     :0  =
-2.50E-01-                           .    0=
-1.25E-01-                              . 0
  .00E+00-                                0
 1.25E-01-                                0  .
 2.50E-01-                                0=    .
 3.75E-01-                                0: =     .
 5.00E-01-                                0* : =     .
 6.25E-01-                                 0*+ :  =     .
 7.50E-01-                                    0*+ :   =    .
 8.75E-01-                                         0* +: =   .
 1.00E+00-                                                   0
 1.12E+00-                                                      . =  :
 1.25E+00-                                                        .
 1.37E+00-                                                          .
 1.50E+00-                                                            .

USED      .94 UNITS
```

Bezüglich der Laufzeit ist der in RAPLØT6 verwendete Algorithmus nicht optimal. Operationen mit alphamerischen Variablen sind eben sehr zeitaufwendig.

Der in RAPLØT6 verwendete Algorithmus bringt allerdings beachtliche Vorteile bei Anwendung für das Zeichnen in Polarkoordinaten (RAPØM). Es sind weniger Speicherplätze erforderlich und damit ist eine Ausnützung aller 72 Druckpositionen in der horizontalen Richtung möglich.

9.10 Eine Aufgabe

Zum Abschluß eine Aufgabe: Es soll der arctan x mit einer beliebigen Methode im DØUBLE PRECISIØN – Modus berechnet werden. Der so erhaltene Wert soll dem Wert den die eingebaute Funktion ATN(X) liefert gegenübergestellt werden. Unten eine Ausgabe eines derartigen Programms. (Programm verwendet Kettenbruchdarstellung)

```
RATN          13:14          01/22/74

     X                 ARCTAN MIT KBR                 ARCTAN MIT ATN

 1.00000E-12    9.99999999999999999 6E-13    1.000000004000678499E-12
 1.00000E-08    9.99999999999999999 663E-09    1.000000004000678466E-08
 1.00000E-04    9.99999996666666666 62E-05    1.000000000667345159E-04
 1.00000E+00    7.85398163397448 3100E-01    7.85398163397448279 0E-01
 1.00000E+04    1.57069632679 52299 52E+00    1.57069632679482988 4E+00
 1.00000E+08    1.57079631679 489 6619E+00    1.57079631679489 6579E+00
 1.00000E+12    1.57079632679 389 6618E+00    1.57079632679389 6619E+00
OUT OF DATA IN 120

USED      .24 UNITS


TAP
READY
130DATA.267949192431122706 5,.577350269189 6257645
131DATA1.7320508075688772 94,-3.7320508075688772 94
RUN-1

RATN          13:16          01/22/74

     X                 ARCTAN MIT KBR                 ARCTAN MIT ATN

 2.67949E-01    2.61799387799149 4363E-01    2.61799387799149 3423E-01
 5.77350E-01    5.23598775598298 8728E-01    5.23598775598298 7812E-01
 1.73205E+00    1.04719755119 659 7746E+00    1.04719755119 659 7733E+00
-3.73205E+00   -1.30899693899 574718 2E+00   -1.30899693899 574709 4E+00
OUT OF DATA IN 120

USED      .28 UNITS
```

Der erste Lauf gibt arctan-Werte für einen weiten Bereich von x. Der zweite Lauf verwendet die x-Werte: $2-\sqrt{2}, \sqrt{3}/3, \sqrt{3}$ und $-(2+\sqrt{3})$, entsprechend der Eingabe 130 und 131, denn die Maske für X in der Ausgabe ist zu klein um 19stellige Zahlen ausgeben zu können. Der arctan für diese speziellen x-Werte ist: $\pi/12$, $\pi/6$, $\pi/3$ und $-5\,\pi/12$. (arctan $1 = \pi/4$ ist im ersten Lauf enthalten)[1]. Die 19te Stelle ist wegen des Abschneidefehlers natürlich mit einer Unsicherheit behaftet, aber die Ausgabe unter KBR ist auf 18 Stellen (gerundet) exakt.

---

[1] Siehe 10 Anhang für Werte der Konstanten.

10. Anhang zu Mark III-BASIC

10.1 ASCII-Code (dezimal) auszugsweise

		(	40	2	50	<	60	F	70	P	80	Z	90
		)	41	3	51	=	61	G	71	Q	81	[	91
	32	*	42	4	52	>	62	H	72	R	82	\	92
!	33	+	43	5	53	?	63	I	73	S	83	]	93
"	34	,	44	6	54	@	64	J	74	T	84	↑	94
#	35	-	45	7	55	A	65	K	75	U	85	←	95
$	36	.	46	8	56	B	66	L	76	V	86		
%	37	/	47	9	57	C	67	M	77	W	87		
&	38	0	48	:	58	D	68	N	78	X	88		
'	39	1	49	;	59	E	69	O	79	Y	89		

10.2 Einige Konstanten auf 19 signifikante Stellen

π	3.14159 26535 89793 238
2π	6.28318 53071 79586 477
$\pi/2$	1.57079 63267 94896 619
$\pi/4$	0.78539 81633 97448 3096
$\sqrt{2\pi}$	2.50662 82746 31000 502
$\sqrt{2}$	1.41421 35623 73095 049
$\sqrt{3}$	1.73205 08075 68877 294
$\sqrt{10}$	3.16227 76601 68379 332
$\sqrt[3]{10}$	2.15443 46900 31883 722
e	2.71828 18284 59045 235
lge	0.43429 44819 03251 8276
1 rad	57.29577 95130 82320 88
1°	1.74532 92519 94329 577E−2
ln2	0.69314 71805 59945 3094
(=1/lde)	

Im Betriebssystem Mark III von General Electric:

Größte Zahl (unendlich):	± 1.70141 18346 04692 317E+38
Bei ØVERFLØW erzeugte Zahl:	± 1.70141 18092 51680 312E+38
Kleinste Zahl ≠ 0 (Rechnerauflösungsvermögen):	± 1.46936 79385 27859 385E−39
Bei UNDERFLØW erzeugte Zahl:	0

Größte ganze Zahl

Einfache Genauigkeit: ± 134 217 727 = $2^{27} - 1$

Erhöhte Genauigkeit: ± 9 223 372 036 854 775 807 = $2^{63} - 1$

11. Index aller Programme

Gesamtübersicht und Kurzbeschreibung aller Programme

Erste Seitenzahl Programm, zweite Seitenzahl zugehörige Ausgabe, wenn getrennt.

RA#1	Anwendung von alphamerischen Variablen	15
RA#2	Anwendung von alphamerischen Variablen	16
RA#3A	Anwendung von alphamerischen Variablen	18
RA#F1	Anwendung von PRINTUSING	23
RA#F2	Anwendung von PRINTUSING	23
RA#F3	Anwendung von PRINTUSING	24
RA#Y	Anwendung von PRINTUSING	25
RA#NA	Beispiel für mehrdimensionale Variable	30
RA#G	Ausgabe für Rekursionsformel für sin und cos	34
RA#D	Zuordnung von Normwerten	36
RA#E	Anwendung einer Iterationsformel (Quadratwurzel)	38
RA#K	Erzeugung Gaußscher Zufallszahlen	41
RA#S	Zeichnen einer Kurve (lineare Skalen)	44
RA#A	Zeichnen einer Kurve (logarithmische Skalen)	47
RAPLO6	Zeichnen von maximal 6 Kurven gleichzeitig	50/49
RAPOM	Zeichnen in Polarkoordinaten	53/55
RA#C	Rechtwinklige Dreiecke mit ganzzahligen Seitenlängen	71/56
RA#Ø1	Ausdrucken von Zahlen in spezieller Anordnung	72/57
RA#U	Gruppierung von Werten in Tabellen	74/58
RA#O	Anwendung alphamerischer Variablen	75/59
RA#M	Ausdrucken von Tabellenköpfen	75/59
RA#W	Anwendung von PRINTUSING und alphamerischer Variablen	76/60
RA#X	Anwendung von PRINTUSING	76/61
RA#P	Ausdrucken eines Schachbrettmusters	77/61
RA#I	Umwandlung pos. ganzer Zahlen in röm. Zahlen	78/64
RA#F	Transformation von kartes. Koord. in Polarkoord.	80/65
RA#LA	Umwandlung von Radians in Grad, Minuten und Sekunden	81/66
RA#J	Ausdrucken des Pascal-Dreiecks	81/66
RA#V	Verallgemeinerte Fibonacci-Reihe	82/67
RA#B1	Berechnung von π	84/68
RA#H	Berechnung einer Differenzentafel	85/70
RA#ØØ3	Beispiel einer Dreiwegverzweigung	72
RA#ØØ4	Beispiel einer Dreiwegverzweigung	73
RA#Z	Tabelle für x, x^2, x^3, $\sqrt{x}$, $\sqrt[3]{x}$, ln(x), lg(x), e^x, 1/x	79
RAPU	Demonstration von PRINTUSING und Zeilenmasken	99
RAMA	Demonstration der MARGIN-Anweisung	100/101
RADICE	Werfen eines Würfels (Zufallszahlengenerator)	101/102

RANØRM	Zuordnung von Normwerten (Unterprogrammtechnik)	103/104
RASØRT	Sortieren numerischer Daten (Unterprogrammtechnik)	105
RAMAT	Lösung eines linearen Gleichungssystems (MAT-Anweisungen)	107
RADYDX	Numerisches Differentieren (DØUBLE PRECISIØN)	109/110
RASIMP	Numerisches Integrieren mit Simpson-Regel	115
RAPLØT6	Gleichzeitiges Zeichnen mehrerer Funktionen (Operationen mit alphamerischen Größen)	118/120

Diese Buchreihe verhilft zu einer schnellen und sicheren Aneignung von Wissen, das für den beruflichen Aufstieg in der heutigen Leistungsgesellschaft erforderlich ist.
An Hand von vollständig durchgeführten Musterbeispielen dringt der Leser schrittweise in das Wissensgebiet ein. Viele Übungen mit angegebenen Lösungen geben ihm die Möglichkeit, die erworbenen Kenntnisse zu überprüfen und schon bald die ihm aufgetragenen Arbeiten zu bewältigen.

Dieses Buch wendet sich, als Folgeband zu Schärf, BASIC für Anfänger, an Leser, die ihre Programmierkenntnisse erweitern, vertiefen und verfeinern wollen. Das Leitmotiv des Buches ist: „Wie programmiere ich besser?": So wird Mark I-BASIC durch Einführung alphamerischer Größen und des PRINTUSING-Befehls, eines äußerst vielseitigen Ausgabe-Befehls, erweitert.
Ein neuer, Mark III-BASIC gewidmeter Teil bespricht Unterschiede zu Mark I-BASIC, führt neue, an alphamerischen Größen operierende Befehle ein und zeigt verschiedene Syntax-Feinheiten auf.
Übungsfragen mit Lösungen, zahlreiche ausgearbeitete Musterprogramme — sowohl in Mark I als auch in Mark III — mit ausführlicher Programmbesprechung und meist für direkte praktische Anwendung sowie viele Aufgaben mit Lösungen machen das Buch geeignet für ein Selbststudium.

FORTBILDUNG DURCH SELBSTSTUDIUM

R. Oldenbourg Verlag Wien München

ISBN 3-7029-0072-1 R. Oldenbourg Wien
ISBN 3-486-39222-5 R. Oldenbourg München